NASA CONTRACTOR REPORT

NASA CR-2415

NASA CR-2415

SEPARATED FLOW OVER BODIES OF REVOLUTION USING AN UNSTEADY DISCRETE-VORTICITY CROSS WAKE

Part II - Computer Program Description

by F. J. Marshall and F. D. Deffenbaugh

Prepared by
PURDUE UNIVERSITY
West Lafayette, Ind.
for Langley Research Center

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • JUNE 1974

1. Report No. NASA CR-2415	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle SEPARATED FLOW OVER BODIES OF REVOLUTION USING AN UNSTEADY DISCRETE-VORTICITY CROSS WAKE - PART II COMPUTER PROGRAM DESCRIPTION		5. Report Date June 1974
		6. Performing Organization Code
7. Author(s) F.J. Marshall and F.D. Deffenbaugh		8. Performing Organization Report No.
		10. Work Unit No. 760-65-11-0200
9. Performing Organization Name and Address Purdue University School of Aeronautics and Astronautics W. Lafayette, Indiana		11. Contract or Grant No. NGR 15-005-119 (181)
		13. Type of Report and Period Covered Contractor Report
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546		14. Sponsoring Agency Code

15. Supplementary Notes

Topical report.

16. Abstract

A method is developed to determine the flow field of a body of revolution in separated flow. The technique employed is the use of the computer to integrate various solutions and solution properties of the sub-flow fields which made up the entire flow field without resorting to a finite difference solution to the complete Navier-Stokes equations. The technique entails the use of the unsteady cross flow analogy and a new solution to the required two-dimensional unsteady separated flow problem based upon an unsteady, discrete-vorticity wake. Data for the forces and moments on aerodynamic bodies at low speeds and high angle of attack (outside the range of linear inviscid theories) such that the flow is substantially separated are produced which compare well with experimental data. In addition, three dimensional steady separated regions and wake vortex patterns are determined.

This report describes the computer program developed to perform the numerical calculations.

17. Key Words (Suggested by Author(s)) Bodies of Revolution, Separated flow, Unsteady flow, Crossflow, Flow field	18. Distribution Statement Unclassified STAR Category 12

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	154	$5.00

*For sale by the National Technical Information Service, Springfield, Virginia 22151

TABLE OF CONTENTS

INTRODUCTION

In part 1 of this two part report the theory was developed to determine the forces acting on a body of revolution in a uniform stream. The body was inclined at an angle of attack sufficient for the flow to become separated and for vortices to form on the lee side. This part of the report describes the computer programs, VCF and ADYNF, which carry out the numerical calculations of part 1. VCF is the main body of the work which calculates the sectional normal force coefficient, c_n, at stations along the body axis. ADYNF using the results obtained from VCF numerically integrates $c_n(\hat{z})$ over the body length to obtain the normal force coefficient C_N, and the moment coefficient $C_{M,\lambda}$. Useful vortex patterns are available, and a line of separation along the body can be determined.

Abbreviations and Additional List of Symbols

Abbreviations

3DS	three dimensional steady
2DUS	two dimensional unsteady
b.l.	boundary layer
r.s.l.	rear shear layer
t.b.l.	top boundary layer
b.b.l.	bottom boundary layer
t.r.s.l.	top rear shear layer
b.r.s.l.	bottom rear shear layer

Sub- and Superscript

$(\ )^v$	point vortex contribution
$(\ )^p$	potential flow contribution

Numerical Integration Parameters

$(\theta_i, t_k, \bar{r}_j)$	set of points in $\bar{r}$, t, θ plane $i = 0,1,2 \ldots,$ IDIM $k = 0,1,2 \ldots,$ KFINAL $j = 0,1,2 \ldots,$ JDIM
$u(\theta_i, t_k, \bar{r}_j)$	$(u)^i$ evaluated at point $(\theta_i, t_k, \bar{r}_j)$
$v(\theta_i, t_k, \bar{r}_j)$	$(v)^i$ evaluated at point $(\theta_i, t_k, \bar{r}_j)$

METHOD OF SOLUTION

The forces acting on a body of revolution at high angle of attack are calculated by the two complementary programs VCF, and ADYNF. Program VCF calculates the normal force distribution on the body by first solving the two dimensional unsteady problem of a circular cylinder started impulsively from rest, and then by using the viscous cross flow analogy to relate the cylinder drag to the normal force sectional coefficient. ADYNF then integrates the normal force distribution supplied by VCF to obtain the normal force and moment coefficients. A detailed description of the method is given in part 1 of this report.

VCF PROGRAM DESCRIPTION

The computer program was written in FORTRAN IV language, and was run on a CDC 6500 computer at Purdue University under the MACE operating system. VCF requires approximately 110000 octal words central memory initially, and CP time for an average run is approximately 30 minutes. The program requires three peripheral disc files in addition to the input, output, and punch files.

VCF PROGRAM INPUT DATA

The input to VCF consists of the Reynolds number based on body length, the angle of attack, the body geometry, and program control data.

Description of Input Decks

The user must supply the body geometry in the form of a FUNCTION subprogram named RZERO. The value of the dimensional body radius r_o^*, should be returned as a function of dimensional axial distance along the body, $\hat{z}^*$. In addition ENTRY DRZERO must return $dr_o^*/d\hat{z}^*$. FUNCTION RZERO is a subprogram deck of VCF supplied by the user.

<u>Deck Setup</u>:

```
    FUNCTION RZERO (ZSTAR)                         CARD 1

    RZERO = (body radius at ZSTAR)                 CARD 2

    RETURN                                         CARD 3
```

ENTRY DRZERO	CARD 4
RZERO = (rate of change of body radius with respect to axial distance evaluated at ZSTAR)	CARD 5
RETURN	CARD 6
END	CARD 7

Description of Input Cards

CARD 1 - Identification. - Card 1 contains any desired identifying information in columns 1-80.

CARD 2 - Flow parameters, body length. - Card 2 contains 3 real numbers, punched in a 12-column field. Columns 37-80 may be used in any desired manner. Card 2 contains the following.

Columns	Variable	Value	Description
1-12	AATACK		Angle of attack in degrees
13-24	RE		$Re_{3DS} = V\ell/\nu$
25-36	LENGTH		Dimensional body length, ℓ, same units as r_o^*.

CARD 3 - Remaining input data. - Card 3 contains three parameters particular to the finite difference and outer flow numerical computation schemes. They are punched in a 12-column field. Columns 37-80 may be used in any desired manner. Card 3 contains the following.

Columns	Variable	Value	Description
1-12	DELT	.125	The time increment used in the finite difference solution to the boundary layer equations, Δt_k
13-24	RC	.05	Potential vortex core cutoff radius, r_c
25-36	SIGMA	.05-1.0	Empirical vortex flux factor, σ

CARD 4 - Program control - The three parameters punched in Card 4, the first right justified in a three column field, and the rest in twelve column fields, determine when the program is to STOP. All data necessary for further execution of the program is written on an auxiliary disk file. Card 4 contains the following.

Columns	Variable	Value	Description
1-3	KFINAL	1-100	If KFINAL = K, where K is an integer count of the number of program time cycles completed, appropriate data is written on an auxiliary disk file and the program STOPS.
13-24	TFINAL		If TFINAL equals or exceeds the central processor time used, data is written on file and the program STOPS.
25-36	ZFINAL	0.0-1.0	If the nondimensional distance along the body axis $\hat{z}$, equals or exceeds ZFINAL data is written on file and the program STOPS.

CARD 5 - Input/Output Control. - Card 5 contains 4 numbers each punched right justified in a 2-column field. These parameters specify which auxiliary disk files are to be used in input/output operations, the type of punched output, and the type of printed output. Card 5 contains the following.

Columns	Variable	Value	Description
1-2	LR	0,3,4	The continue data is READ from TAPE (LR). If LR = 0 no data is read from the auxiliary disk fil
3-4	LW	3-4	The continue data is WRITTEN on TAPE (LW).
5-6	LEVEL	5	Printed Output Level.
7-8	KPUN		Cards are punched every KPUN cycles.

VCF PROGRAM OUTPUT DATA

Printed output is processed by a standard 132 characters-per-line printer, punched cards are in Hollerith (026) code, 80 characters per cards. Binary coded data is written on auxiliary files, TAPE 1, TAPE 3, and TAPE 4. The program printed output options are described below:

LEVEL = 3	The program prints the dimensional body geometry as the radius as a function of axial distance. The maximum body diameter, the characteristic length, the fineness ratio, and frontal area, which are all functions of input body geometry are printed. Remaining card input data is printed. Point vortex locations, velocities and strengths, are printed. The pressure distribution around the two-dimensional cylinder corresponding to an axial station along the three dimensional body is printed for each program cycle, as well as values of shear and pressure drag. Data specifying the rear separation angle is printed for each cycle after the backflow velocity has exceeded .1 of the free stream velocity. Elapsed computer execution time is printed at the end of each program cycle.
LEVEL = 5	In addition to level 3 output, the boundary layer profile for the top portion of the cylinder is printed.

The program printed output is illustrated by the sample case presented in Appendix IV.

The program punched output consists of variables punched every program cycle, and of variables punched every KPUN cycles. Values of nondimensional axial distance, coefficient of drag, nondimensional time, and the number of time cycles completed are punched every cycle. Vortex positions, velocities, and strengths are punched every KPUN cycles.

The binary coded data output consists of initial boundary layer velocities, (reference 2), written on TAPE 1. Values of boundary layer velocities, vortex positions, vortex velocities, vortex strengths, program indices, and rear shear layer data are written on TAPE (LW) at the termination of each run.

VCF PROGRAM STRUCTURE

VCF consists of a MAIN program, 18 SUBROUTINE subprograms, 5 FUNCTION subprograms, and 1 FUNCTION subprogram supplied by the user. Detailed descriptions of the MAIN program and of the subprograms are given in Appendix I. An example of a user supplied geometry description deck is given in Appendix IV for the case of an ogive cylinder.

VCF AUXILIARY FILES

VCF designates TAPE 5 as the input file, TAPE 6 as the output file, and TAPE 7 as the punch file. In addition three auxiliary files are utilized for transfer of binary data. These files are designated TAPE 1, TAPE 3, and TAPE 4.

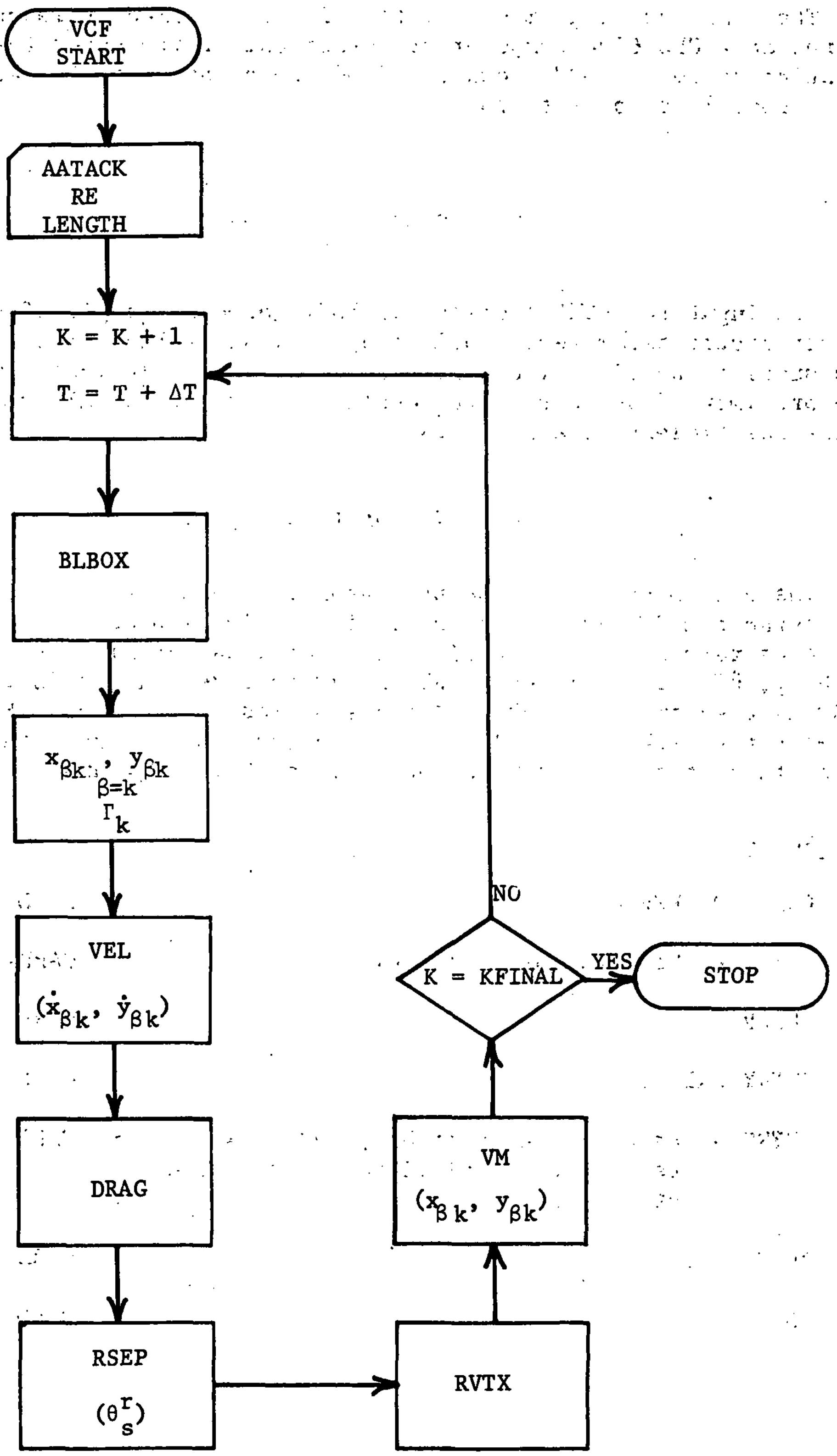
TAPE 1 is used to store the initial boundary layer profile. TAPE 3 and TAPE 4 are used to store intermediate information which is used to continue execution of the program.

These auxiliary files must be rewound and stored on magnetic tape or some other permanent storage file at the termination of each run. To continue the run the files are obtained from permanent file storage, and execution is continued, the necessary data being READ from TAPE 3 or TAPE 4.

VCF OPERATING INSTRUCTIONS

The program deck, geometry input deck, and data deck are loaded in the following sequence: job card, system control cards, end-of-record card, program deck, geometry input deck, end-of-record card, data deck, end-of-file card. The geometry input deck and the data deck are described in the Program Input Data section.

VCF FLOWCHART



ADYNF PROGRAM DESCRIPTION

The computer program was written in the FORTRAN IV language, and was run on a CDC 6500 computer at Purdue University under the MACE operating system. ADYNF requires 55000 words of central memory and takes about 10 seconds to run.

ADYNF PROGRAM INPUT DATA

The input to ADYNF consists of body geometry, angle of attack, characteristic parameters supplied by VCF, and values of c_n and $\hat{z}$, also supplied by VCF. The number of knots and their locations, used to approximate functions in the evaluation of the normal force and moment coefficients are also read in.

Description of Input Decks

The user must supply the body geometry in the form of a FUNCTION subprogram named RZERO. The value of the dimensional body radius r_o^*, should be returned as a function of dimensional axial distance along the body, $\hat{z}^*$. In addition ENTRY DRZERO must return $dr_o^*/d\hat{z}^*$. FUNCTION RZERO is a subprogram deck of ADYNF supplied by the user. For a given geometry the deck used in ADYNF is identical to that used in PROGRAM VCF, and in fact, the same deck may be used for both programs

<u>Deck Setup</u>:

FUNCTION RZERO (ZSTAR)	CARD 1
RZERO = (body radius at ZSTAR)	CARD 2
RETURN	CARD 3
ENTRY DRZERO	CARD 4
RZERO = (Rate of change of body radius with respect to axial distance evaluated at ZSTAR)	CARD 5
RETURN	CARD 6
END	CARD 7

Description of Input Cards

CARD 1 - Identification. - Card 1 contains any desired identifying information in columns 1-80.

CARD 2 - Flow parameters, characteristic dimensions. - Card 2 contains 6 real numbers, punched in a 12-column field. Columns 73-80 may be used in any desired manner. Card 2 contains the following.

Columns	Variable	Value	Description
1-12	AATACK		Angle of attack in degrees
13-24	LAMBDA		Moment arm, λ
25-36	LENGTH		Dimensional body length, ℓ
37-48	F		Fineness Ratio, f
49-60	AW		Characteristic length, $\tilde{a}$
61-72	RW		Maximum Radius, d/2

CARD 3 - Number of knots, number of data points. - Card 3 contains 2 integers right justified in a 4 column field. The number of knots refers to the number of cubic polynomials used to obtain an approximating function to the data. A cubic polynomial approximates the data between two knots. Card 3 contains the following.

Columns	Variable	Value	Description
1-4	NOKNOT	4-5	The interval is divided into 3 or 4 segments depending on whether NOKNOT is 4 or 5. The endpoints of the segments are the knots.
5-8	LX		The number of data points, $\hat{z}$, $C_D(\hat{z})$.

CARD 4,5 ... - Axial distance $\hat{z}$, Coefficient of drag $C_D(\hat{z})$. - Card 4,5 ... contain values of $\hat{z}$ and $C_D(\hat{z})$, these cards are punched by PROGRAM VCF. The two numbers are punched in the first two 12-column fields of each card.

CARD LAST - Knot positions. - The last card in the input deck to ADYNF contains the location of the knots. Since the nondimensional body length is 1.0, these knot values should be: 0, values contained in the interval [0,1], and 1. The knot values are punched in a 12-column field 6 to a card. The last card contains the following.

Columns	Variable	Value	Description
1-12	XI(1)	0	Location of first knot
13-24	XI(2)	0<( )<1	Location of 2nd knot
.			
.			
.	XI(NOKNOT)	1	Location of last knot

ADYNF PROGRAM OUTPUT DATA

Program ADYNF output is printed output processed by a standard 132 character-per-line printer. The program printed output consist of:

(1) The given data $[\hat{z}, c_n(\hat{z})]$

(2) The number of knots

(3) Initial knot locations

(4) Optimized knot locations (reference 4)

(5) The cubic coefficients used to approximate the data in each interval

(6) The errors involved in approximating the data

(7) The data point $\hat{z}$ and the approximation $c_n(\hat{z})$

(8) Values of $\hat{z}$ at .05 increments from 0 to 1.0 and the value of the approximation $c_n(\hat{z})$

(9) Intermediate output from the integration subroutine CADRE (see Appendix II for details)

(10) The approximation to the integral $\int_0^1 c_n(\hat{z})\, d\hat{z}$. The absolute error, and an indication as to the types of singularities involved in the integration. (see APPENDIX II - CADRE for further details)

(11) The normal force coefficient C_N

(12) Similar output for the calculation of the moment coefficient $C_{M,\lambda}$.

ADYNF PROGRAM STRUCTURE

ADYNF is a main program which references the subprograms RZRO, FIT, SPLINEB, CADRE and the user input geometry subprogram RZERO. SPLINE, references 3 and 4, and CADRE are "canned" routines obtained from Purdue University's computing center. SPLINE is a deck of subprograms. ADYNF calls SPLINEB which references other subprograms in the SPLINE deck. Those subprograms are considered to be "black boxes," and are not explained in detail in this report. However listings of the SPLINE deck are included in APPENDIX II. Listings of ADYNF, FIT, and RZRO are given in APPENDIX III. ADYNF designates TAPE 5 as input file and TAPE 6 as output file. No other files are used.

ADYNF OPERATING INSTRUCTIONS

The program deck, geometry input deck, and data deck are loaded in the following sequence: job card, system control cards, end-of-record card, program deck, geometry input deck, end-of-record card, data deck, end-of-file card. The geometry input deck and the data deck are described in the Program Input Data section.

APPENDIX I

PROGRAM VCF AND SUBPROGRAM DESCRIPTIONS

This appendix contains a brief outline of the purpose, method, and use of program VCF and its subroutines. The principal variables and constants in each are listed, and identified as input or output data. The subprograms are listed in alphabetical order.

PROGRAM VCF

PURPOSE: To calculate the local normal force distribution coefficient $c_n(\hat{z})$, to predict the line of separation $\theta_s(\hat{z})$, and to provide the point vortex distribution, $(x_{\beta k}, y_{\beta k})$.

METHOD: VCF proceeds in discrete steps of time Δt_k, usually given the value .125. The index K, initially 0, is incremented by 1 for each step in time, thus K=1 for $t_1 = t_o + \Delta t_1$, where t_o denotes the initial time. The boundary layer equations are integrated using a finite difference solution developed by M. G. Hall, reference 1. The initial velocity distribution is supplied by Wundt's solution, reference 2, to the impulsive start of a circular cylinder. Once the initial conditions have been calculated the subroutines used in the computations are no longer necessary. By writing all of the pertinent data on a FILE, in this instance either TAPE3 or TAPE4, and then replacing the inert subroutine with a dummy subroutine IC, and reading the data back in, the central memory requirement can be reduced after the first few minutes of computation time. Further, since the boundary layer solution employs a finite difference method, storage must be allocated for a finite difference mesh or grid. However, since separation soon occurs, values at mesh points greater than the separation angle are no longer needed. By stopping execution, saving the necessary data, redimensioning the boundary layer grid, and then reading the data back in, and continuing execution, the central memory requirements can be reduced even further. For the case of the ogive cylinder the central memory requirement could be reduced from 110000 words to 65000 words. The boundary layer grid is DIMENSIONED in the main program and values of u^i, and v^i are passed as parameters to subprograms with the DIMENSION parameters IDIM, JDIM. Thus if boundary layer grid values are no longer being used because they are in separated flow the variables U, W, UT, and UB are appropriately DIMENSIONED in the main program.

The boundary layer finite difference scheme is applied either to the top half of the cylinder or the bottom. The entire boundary layer solution could be accomplished by a call to BLBOX with appropriate values of velocity on the upper surface being supplied, and then another call to BLBOX with lower surface values. However, since the solution is basically a small time one, and the flow is expected to be symmetrical, it is only necessary to apply the finite difference solution to half of the cylinder.

Once the velocity distribution in the boundary layer is known, and a separation angle has been determined, vorticity is introduced into the outer flow in the form of point vortices. Due to the symmetry of the problem separate vortex arrays were created for the vortices born from the top separation angle, and for those coming from the bottom separation angle. In addition when backflow velocities were great enough to generate vorticity on the rear portion of the cylinder two additional top and bottom arrays were created for those point vortices originating from the rear separation angles.

Bernoulli's equation yields the pressure distribution on the cylinder, and numerically integrating $C_p \cos \theta$ $(0 < \theta < 2\pi)$, gives the drag.

The point vortices existing at time t_k are convected with the fluid and at time $t_{k+1} = t_k + \Delta t_k$ the tangential velocity on the surface (boundary conditions for the boundary layer equations), will be altered. A new boundary layer velocity profile is calculated, a new separation angle determined, and the cycle is repeated. When t_k corresponding to $\hat{z} = 1.0$ is reached,the program stops.

USE: Input:

INFO	Data identification
AATACK	Angle of attack
RE	Reynolds number based on body length
LENGTH	Dimensional body length
DELT	Finite difference time step
RC	Vortex core cutoff value
SIGMA	Empirical vortex flux factor
KFINAL	Program cycle termination index
TFINAL	Program central processor time termination value
ZFINAL	Nondimensional length termination value
LR	READ continue data from TAPE(LR) if LR=0 no binary continue data is READ
LW	WRITE continue data on TAPE(LW)
KPUN	Vortex positions, velocities, and strengths are punched every KPUN cycles
LEVEL	Printed output level

Output:

PI	π
DTOR	$\pi/180$
RTOD	$180/\pi$
DMAX	Maximum body diameter
AW	Characteristic length $\tilde{a}$, of the 2DUS flow
F	Fineness ratio, f
SA	Frontal area, S
RE	Re_{2DUS}
KS	Initially equal to 1000, KS is set equal to K when the first point vortex is born from the b.l.
KR	Initially equal to 1000, KR is set equal to K when the backflow velocity exceeds a value of .1
KT	Equal to the number of t.b.l. vortices in the flow at time t_k
KB	Equal to the number of b.b.l. vortices in the flow at time t_k
KRT	Equal to the number of t.r.s.l. vortices in the flow at time t_k
KRB	Equal to the number of b.r.s.l. vortices in the flow at time t_k
INITAL	The initial dimension of the θ grid in the b.l.
NDIM	Dimension of the vortex arrays, and of the drag array. The number of program cycles must not exceed NDIM
IDIM, JDIM	b.l. grid dimension $(\theta_i, \bar{r}_j)$ $i = 1, 2, \ldots,$ IDIM $j = 1, 2, \ldots,$ JDIM
PI2	2π
SQRTPI	$\sqrt{\pi}$

SQA	Square of the sine of the angle of attack
SRE	$\sqrt{RE_{2DUS}}$

ISYM	Symmetric mode index
	ISYM = 1, symmetric mode
	ISYM = 0, asymmetric mode

THTASYM	Condition on θ_s, if THTASYM is greater than θ_s, the program goes into the asymmetric mode
K	Program time index, initially set equal to 0 at t_o. K is incremented by 1 for each successive step in time
KTS	Index used in the continuation of a run, if KTS $\geq$ K the b.l. grid is calculated
T	t_k
TI	t_o
ZHAT	$\hat{z}$
AK	Radius at time t_k, $a(t_k)$
AKTI	$a(t_o)$
AKDOT	$\dot{a}(t_k)$
CDPI	Initial drag coefficient due to pressure = $2\pi\ a(t_o)\ \dot{a}(t_o)$
KMINUS1	K-1
TH	$t_{k-1/2}$
AK	Radius at time t_k, $a(t_k)$
AKDOT	$\dot{a}(t_k)$
AADOT	$a(t_k)\ \dot{a}(t_k)$
AKSQD	$[a(t_k)]^2$
ZHALF	$\hat{z}(t_{k-1/2})$
AKPHALF	$a(t_{k-1/2})$
AKDOTPH	$\dot{a}(t_{k-1/2})$
KTTEMP	Temporary storage of the value of KT

KBTEMP	Temporary storage of the value of KB
U	Boundary layer velocity, $\overline{u}(\overline{r},t_k,\theta)$
DGAMMA	$(\Gamma)_{,t}$
SMALLM	m_k
CDST	$\int_0^{\theta_s} \tau \sin\theta \, d\theta \quad 0 < \theta_s < \pi$
CLST	$\int_0^{\theta_s} \tau \cos\theta \, d\theta \quad 0 < \theta_s < \pi$
ISEP	An integer denoting the element in the θ_i array, the value of which is the separation angle
TRAPD	Trapezoidal sum used in the evaluation of CDST and CDSB
TRAPL	Trapezoidal sum used in the evaluation of CLST and CLSB
CDSK	Viscous contribution to the drag coefficient = $\frac{1}{2}\int_0^{2\pi} \tau \sin\theta \, a(t) \, d\theta$
TT	Array of t.b.l. vortex diffusion times
TB	Array of b.b.l. vortex diffusion times
TRT	Array of t.r.s.l. vortex diffusion times
TRB	Array of b.r.s.l. vortex diffusion times
X, Y	Arrays of t.b.l. vortex coordinates
XDOT, YDOT	Arrays of t.b.l. vortex velocities $(\tilde{u}^\circ, \tilde{v}^\circ)$
GAMMA	Array of t.b.l. vortex strengths
XB, YB	Arrays of b.b.l. vortex coordinates
XDOTB, YDOTB	Arrays of b.b.l. vortex velocities

GMAB	Array of b.b.l. vortex strengths
XRT, YRT	Arrays of t.r.s.l. vortex coordinates
XRDOT, YRDOT	Arrays of t.r.s.l. vortex velocities
GMRT	Array of t.r.s.l. vortex strengths
XRB, YRB	Arrays of b.r.s.l. vortex coordinates
XRDOTB, YRDOTB	Arrays of b.r.s.l. vortex velocities
GMRB	Array of b.r.s.l. vortex strengths
CDSB	$\int_0^{\theta_s} \tau \sin\theta \, d\theta \quad \pi < \theta_s < 2\pi$
CLSB	$\int_0^{\theta_s} \tau \cos\theta \, d\theta \quad \pi < \theta_s < 2\pi$
THETA	θ
THETAS	$\theta_s \quad 0 < \theta < \pi$
THETASB	$\theta_s \quad \pi < \theta < 2\pi$
THTASR	θ_s^r
IPUN	Punch index IPUN ≥ 1, punch vortex arrays IPUN < 1, no vortex array punch
CDPK	Drag coefficient due to pressure = $\frac{1}{2}\int_0^{2\pi} C_p \cos\theta \, a(t) \, d\theta$
CDK	Drag coefficient, $C_D(t)$
CDN	Local normal force distribution coefficient, c_n
TIME	Execution time

SUBPROGRAMS CALLED: BLBOX, DQI, DRZRO, NONDIM, PDRAG, RSEP, RVTX, RZRO, SECOND, VEL, VM, WRIT

FUNCTION AN

PURPOSE:	To calculate the coefficients a_n, b_n, c_n
METHOD:	Given the b.l. velocity distribution the coefficients a_n, b_n, c_n are calculated as outlined in reference 1.
USE:	Function reference to

AN (L, N, IDIM, JDIM, U, W)

Input:

L, N	The b.l. grid point (θ_i, $\bar{r}_j$)
IDIM, JDIM	b.l. grid dimension (θ_i, $\bar{r}_j$) $i = 1,2, \ldots,$ IDIM $j = 1,2, \ldots,$ JDIM
U	$u(\theta_i, t_k, \bar{r}_j)$
W	$u(\theta_i, t_k, \bar{r}_j)$
DS	
DZ8	
DZSQ2	See BLBOX
DZSQ4	

Output:

AN	Coefficient a_n in the finite difference solution of the b.l. equations

CALLED FROM:	BLBOX
REMARKS:	Alternate ENTRY points to AN are BN, and CN

SUBROUTINE BLBOX

PURPOSE:	To numerically integrate the unsteady b.l. equations
METHOD:	A finite difference method is employed to solve the two dimensional unsteady boundary layer equations, (reference 1).
USE:	CALL BLBOX (SMALL M, DGAMMA, TAWD, TAWL, U, W, UT, UB, IDIM, JDIM, MODE)

<u>Input</u>:

U	The b.l. velocity distribution $u(\theta_i, t_{k-1}, \bar{r}_j)$
UT	2-dimensional storage away for $u(\theta_i, t_{k-1}, \bar{r}_j)$
UB	2-dimensional storage away for $u(\theta_i, t_{k-1}, \bar{r}_j)$ $\pi < \theta_i < 2\pi$
UTNBIG	An array the elements of which are the boundary conditions (t.b.l.), for the finite difference scheme
UBNBIG	Boundary conditions (b.b.l.)
IDIM, JDIM	b.l. grid dimension $(\theta_i, \bar{r}_j)$ $i = 1,2, \ldots,$ IDIM $j = 1,2, \ldots,$ JDIM
MODE	If MODE = 1 appropriate t.b.l. u^i values will be input to the b.l. grid. If MODE = 2 b.b.l. u^i values are input
KTS	Index used in the continuation of a run, if KTS $\geq$ K the b.l. grid is calculated
AK	$a(t_k)$
AKSQD	$[a(t_k)]^2$
AKDOT	$\dot{a}(t_k)$
AKPHALF	$a(t_{k-1/2})$
IXTRSET	An integer denoting the element in the S array, the value of which is the t.b.l. separation angle

IXBRSET	An integer denoting the element in the S array, the value of which is the b.b.l. separation angle
NBIG	An integer defining the last element in the AN array

<u>Output</u>:

DELS	$\Delta\theta$ used to define the b.l. grid $\theta_{i+1} = \theta_i + \Delta\theta$
DS	Defining the elements of the array DS as, $\Delta\theta_{i+1} = \theta_{i+1} - \theta_i$, allows for the grid defined to have a variable mesh spacing.
S	Array of θ_i values
ST	Array of t.b.l. θ_i values
SB	Array of b.b.l. θ_i values
DZ	b.l. grid spacing $\Delta\bar{r} = .14$
DZ2	$1/2\ \Delta\bar{r}$
DZSQ2	$1/2\ \Delta\bar{r}^2$
DZSQ4	$1/4\ \Delta\bar{r}^2$
ZN	Array of $\bar{r}$ values
U	The b.l. velocity distribution $u(\theta_i, t_k, \bar{r}_j)$
UT	2-dimensional storage array for $u(\theta_i, t_k, \bar{r}_j)$ $\pi < \theta_i < 2\pi$
UB	2-dimensional storage array for $u(\theta_i, t_k, \bar{r}_j)$ $\pi < \theta_i < 2\pi$
TAUS TAUNEW	Successive estimates of the surface shear τ
NCYCLE	The number of iterative cycles, (reference 1)
DUDR	$(\bar{u})_{,\bar{r}}$
TAW	τ
TAWD	$\tau\ \sin\theta$ evaluated for values of the θ grid
TAWL	$\tau\ \cos\theta$ evaluated for values of the θ grid

THETA	θ	
THETAS	θ_s	$0 < \theta < \pi$
THETASB	θ_s	$\pi < \theta < 2\pi$
SMALLM	Location of vortex born from b.l., m_k	
DGAMMA	$(\Gamma)_{,t}$	

CALLED FROM: VCF

SUBPROGRAMS CALLED: IC, FREEVTX, POTFLOW, TRID, RITE

REMARKS: The b.l. subroutine determines the velocity profile for $0 < \theta < \theta_s$ $(0 < \theta_s < \pi)$, or for $0 < \theta < \theta_s$ $(\pi < \theta_s < 2\pi)$. Thus if the flow is assumed to be symmetric, a finite difference solution is required only on the top half of the cylinder. If the flow is asymmetric, BLBOX is called twice with appropriate top or bottom grid values and boundary conditions being input.

FUNCTION CPC

PURPOSE:	To calculate C_p
METHOD:	The equations for evaluating C_p are given in part 1
USE:	Function reference to
	CPC (THTA)

<u>Input</u>:

TT	
TB	
TRT	
TRB	
X, Y	
XDOT, YDOT	
GAMMA	See VCF
XB, YB	
XDOTB, YDOTB	
GMAB	
XRT, YRT	
XRDOT, YRDOT	
GMRT	
XRB, YRB	
XRDOTB, YRDOTB	
GMRB	

K	
KT	
KB	See VCF
KRT	
KRB	
XX	X coordinate of the point on the cylinder surface at which C_p is to be evaluated
YY	Y coordinate of the point on the cylinder surface at which C_p is to be evaluated
NDIM	Dimension of the vortex arrays

<u>Output</u>:

PHIT	$\Phi,_t = (\Phi,_t)^v + (\Phi,_t)^p$
PHIVT	$(\Phi,_t)^v$
PHIPT	$(\Phi,_t)^p$
PSIKR2	$\overset{\circ}{u_\circ}$
CPC	Pressure coefficient

CALLED FROM: VCF

SUBPROGRAMS CALLED: PV, FREEVTX, POTFLOW

REMARKS: The time derivative of the potential function, $(\Phi,_t)$, see part 1, is computed by first determining the terms due to the point vortices $(\Phi,_t)^v$, adding to that the potential flow solution of a changing radius cylinder in a uniform flow $(\Phi,_t)^p$.

FUNCTION DN

PURPOSE:	To determine the coefficient d_n, (reference 1)
METHOD:	The equations for the calculation of d_n are given in reference 1
USE:	Function reference to

DN (L, N, IDIM, JDIM, U, W)

Input:

L, N	The b.l. grid point $(\theta_i, \bar{r}_j)$
IDIM, JDIM	b.l. grid dimension $(\theta_i, \bar{r}_j)$ $i = 1,2, \ldots,$ IDIM $j = 1,2, \ldots,$ JDIM
U	$u(\theta_i, t_k, \bar{r}_j)$
W	$v(\theta_i, t_k, \bar{r}_j)$
DS	
DZ2	see BLBOX
NBIG	
DT2	$1/2\ \Delta t_k$

Output:

DN	Coefficient d_n in the finite difference solution of the b.l. equations

CALLED FROM:	BLBOX
SUBPROGRAMS CALLED:	UM, US, UL, AN

SUBROUTINE DQI

PURPOSE: To evaluate $\int_0^{2\pi} C_p \cos\theta \, d\theta$

METHOD: Numerical integration using Simpsons rule

$$\int_{X_o}^{X_2} f(x)dr \sim \frac{h}{3}\left[f(x_2) + 4f(x_1) + f(x_o)\right]$$

USE: CALL DQI (FCT, CK, K, N, ISYM)

<u>Input</u>:

FCT	The name of the EXTERNAL FUNCTION SUBPROGRAM used; PDRAG to evaluate drag, PLIFT to evaluate lift.
K	Program time index, see VCF
N	An even integer which determines the integration stepsize h. The front of the cylinder is divided into 2N equal parts, the front here being $-60° < \theta < 60°$, and the back is divided into 8N equal parts. The stepsize, $h = (60/N)(\pi/180)$.
ISYM	An integer either 0 or 1. A value of 1 denotes a symmetric flow, and the limits of integration are $(0,\pi)$ and the value of the integral is doubled. A value of 0 denotes an asymmetric flow and the limits of integration are $(0,2\pi)$

<u>Output</u>:

CK	Approximation to $\int_0^{2\pi} C_p \cos\theta \, d\theta$ by Simpson's Rule

CALLED FROM: VCF

SUBPROGRAMS CALLED: PDRAG

REMARKS: The value of N used in all testing was 6.

SUBROUTINE FREEVTX

PURPOSE:	To calculate vortex induced velocities
METHOD:	Each point vortex and its image induce a velocity at every other point in the field. The distance between the field point and the vortex location is determined and induced velocity calculated, see part 1. Summing the effects of all the point vortices yields the vortex induced velocity.
USE:	CALL FREEVTX (PSIK1X, PSIK1Y, PSIK1R, IA)

<u>Input</u>:

IA	An integer value from 1 to 5. IA = 1 Field point is a t.b.l. vortex IA = 2 Field point is a b.b.l. vortex IA = 3 Field point is on the cylinder IA = 4 Field point is a t.r.s.l. vortex IA = 5 Field point is a b.r.s.l. vortex
TT	
TB	
TRT	
TRB	
X, Y	
XDOT, YDOT	See VCF
GAMMA	
XB, YB	
XDOTB, YDOTB	
GMAB	
XRT, YRT	

XRDOT, YRDOT	See VCF
GMRT	
XRB, YRB	
XRDOTB, YRDOTB	
GMRB	
KT	
KB	
TRT	
KRB	
ALPHA	An integer denoting a particular point vortex, the coordinates of which is the point at which the velocity is being calculated.
XX	X coordinate of the point vortex at which the velocity is being calculated.
YY	Y coordinate of the point vortex at which the velocity is being calculated.

<u>Output</u>:

PSIK1X	$(\tilde{v}^\circ)^v$
PSIK1Y	$-(\tilde{u}^\circ)^v$
PSIK1R	$(u^\circ_\circ)^v$

CALLED FROM: BLBOX, VEL, CPC

SUBPROGRAMS CALLED: PSI

SUBROUTINE IC

PURPOSE:	To calculate the initial boundary layer profile for the problem of a circular cylinder started impulsively from rest.
METHOD:	Wundt's solution for the impulsive start of a circular cylinder,(reference 2).
USE:	CALL IC (AKTI, U, IDIM, JDIM, MODE)

Input:

AKTI	$a(t_o)$
IDIM, JDIM	b.l. grid dimension $(\theta_i, \bar{r}_j)$ $i = 1,2 \ldots, \text{IDIM}$ $j = 1,2 \ldots, \text{JDIM}$
MODE	MODE = 1 $0 < \theta_i < \pi$ MODE = 2 $\pi < \theta_i < 2\pi$

Output:

U	$u(\theta_i, t_o, \bar{r}_j)$
UTNBIG	An array, the elements of which are the boundary conditions (t.b.l.) for the finite difference scheme. $u(\theta_i, t_o, \bar{r}_j)$ $j = \text{JDIM}$ $0 < \theta_i < \pi$
UBNBIG	Boundary conditions (b.b.l.) $u(\theta_i, t_o, \bar{r}_j)$ $j = \text{JDIM}$ $\pi < \theta_i < 2\pi$

CALLED FROM:	BLBOX
SUBPROGRAMS CALLED:	PHI0, PHI1, ZTA001, ZTA0011, ZTA02A1, ZTA02B1
REMARKS:	Subroutine IC is called once from BLBOX when $t_k = t_o$ (K=1). A dummy subroutine may be substituted thereafter, reducing the central memory requirement. The IC subroutine and the subprograms called from IC are listed in Appendix Ib.

SUBROUTINE NONDIM

PURPOSE:	To calculate the characteristic radius $\tilde{a}$, the maximum diameter d, and the reference area S, for the input body geometry.
METHOD:	Values of $r_o^*(\hat{z}^*)$ are obtained from RZERO, the FUNCTION SUBPROGRAM input by the user, for values of $\hat{z}^*$ along the body axis. The maximum $r_o^*(\hat{z}^*)$, doubled, is d. For a closed end body $(r_o^*(\ell) = 0)$ the characteristic radius is obtained by using the trapezoidal rule to approximate the integral $1/\ell \int_0^\ell r_o^*(\hat{z}^*)dz^* = \tilde{a}$. For open ended geometries $(r_o^*(\ell) \neq 0)$, then $\tilde{a} = d/2$.
USE:	CALL NONDIM (DMAX, RW, AW, F, S, L, PI)

<u>Input</u>:

L	Dimensional body length, ℓ
PI	π.

<u>Output</u>:

DMAX	Maximum body diameter, d
RW	d/2
AW	Characteristic length, $\tilde{a}$
F	Fineness ratio, ℓ/d
S	Frontal area = $\pi d^2/4$

CALLED FROM:	VCF
SUBPROGRAMS CALLED:	RZERO

FUNCTION PDRAG

PURPOSE:	To calculate $C_p(\theta)\cos\theta$
METHOD:	Function reference to CPC
USE:	Function reference to PDRAG (THETA) <u>Input</u>: THETA θ <u>Output</u>: PDRAG $C_p(\theta)\cos\theta$
CALLED FROM:	VCF
SUBPROGRAMS CALLED:	CPC
REMARKS:	See DQI

SUBROUTINE POTFLOW

PURPOSE:	To calculate the velocity at a point in the outer flow due to the uniform inviscid flow about a circular cylinder.
METHOD:	Evaluate potential flow solution at a field point
USE:	CALL POTFLOW (PSIKXP, PSIKYP, PSIKRP, IA)

Input:

ALPHA	Point vortex index
AKSQD	$[a(t_k)]^2$
IA	An integer value from 1 to 5 IA = 1 Field point is a t.b.l. vortex IA = 2 Field point is a b.b.l. vortex IA = 3 Field point is on the cylinder IA = 4 Field point is a t.r.s.l. vortex IA = 5 Field point is a b.r.s.l. vortex
X, Y	
XDOT, YDOT	
GAMMA	
XB, YB	
XDOTB, YDOTB	See VCF
GMAB	
XRT, YRT	
XRDOT, YRDOT	
GMRT	
XRB, YRB	

XRDOTB, YRDOTB		See VCF
GMRB		

<u>Output</u>:

PSIKXP	$(\tilde{v}^{\circ})^p$
PSIKYP	$-(\tilde{u}^{\circ})^p$
PSIKRP	$(u_{\circ}^{\circ})^p$

CALLED FROM: BLBOX, VEL, CPC

SUBROUTINE PSI

PURPOSE:	To sum the induced vortex velocity
METHOD:	The equations for the induced velocity of a point vortex are given in part 1.
USE:	CALL PSI (X, Y, GMA, NDIM, KI, KF, IA, IB, SUM, SUM1, TK)

<u>Input</u>:

AK	$a(t_k)$
AKSQD	$[a(t_k)]^2$
XX, YY	Coordinates of the point at which the velocity is being calculated
X, Y	Input arrays of point vortex locations
GMA	Input array of point vortex strengths
NDIM	Dimension of the input arrays
KI, KF	Sum over KF - KI + 1 vortices
TK	Input array of vortex diffusion times
RC	r_c
IA	Field point index (See FREEVTX)
IB	Integer which determines if a point vortex is to be ignored in the sum

<u>Output</u>:

SUM	$\tilde{v}°$ induced by input vortex array
SUM1	$-\tilde{u}°$ induced by input vortex array

CALLED FROM: FREEVTX

SUBROUTINE PV

PURPOSE:	To calculate the vortex contribution to C_p
METHOD:	Equations for the calculation of $\Phi,_t$ are given in part 1.
USE:	CALL PV (X, Y, XDT, YDT, GMA, NDIM, KI, KF, SUM, TK)

<u>Input</u>:

AK	$a(t_k)$
AKSQD	$[a(t_k)]^2$
XX, YY	Coordinates of the point on the cylinder at which C_p is to be evaluated
X, Y	Input arrays of point vortex locations
XDT, YDT	Input arrays of point vortex velocities
GMA	Input array of point vortex strengths
NDIM	Dimension of the input arrays
KI, KF	Sum over KF - KI + 1 vortices
TK	Input array of vortex diffusion times
RC	r_c

<u>Output</u>:

SUM	$\Phi,_t$ due to input vortex array

CALLED FROM: CPC

SUBROUTINE RITE

PURPOSE:	To print the b.l. velocity distribution
METHOD:	ENCODE is used to create execution time FORMAT statements, so that only the velocity profile for $0 < \theta < \theta_s$ is printed.
USE:	CALL RITE (MT, IDIM, JDIM, U, ZN, NX, S, DEG, ZHAT, DELS, K)

<u>Input</u>:

MT	An integer value equal to 1 or 2 If MT = 1 the initial velocity profile is printed If MT = 2 the velocity profile at t_k is printed
IDIM, JDIM	b.l. grid dimension $(\theta_i, \bar{r}_j)$ $i = 1,2, \ldots,$ IDIM $j = 1,2, \ldots,$ JDIM
U	$u(\theta_i, t_k, \bar{r}_j)$
ZN	Array of $\bar{r}_j$ values $j = 1,2, \ldots,$ JDIM
NX	An array containing a count of the number of iterative cycles, (reference 1)
S	Array of θ_i values $i = 1,2, \ldots,$ IDIM
DEG	Array of θ_i values in degrees
ZHAT	$\hat{z}$
DELS	$\Delta\theta$
K	Program cycle index

CALLED FROM:	BLBOX, IC
REMARKS:	When MT = 1, u^i is output, when MT = 2 $\bar{u}$ is output.

SUBROUTINE RSEP

PURPOSE:	To determine the r.s.l. separation angle
METHOD:	$\dfrac{\theta_s^f - \theta_m^f}{\theta_o - \theta_m^f} = \dfrac{\theta_m^r - \theta_s^r}{\theta_m^r - \theta_o}$ (see part 1)
USE:	CALL RSEP (THTASF, THTASR, THTA, UBL, INTX1, INITAL, MODE)

<u>Input</u>:

THETASF	b.l. separation angle, θ_s^f
THTA	Array of θ_i values i = 1,2,..., INITAL
UBL	Array of u_o° (θ_i) values i = 1,2, ..., INITAL
INTX1	An integer denoting the element of the THTA array which is the separation angle
INITAL	The inital dimension of the b.l. grid, $(\theta_i = \pi)$ i = INITAL
MODE	MODE = 1 $\quad 0 < \theta_i < \pi$
	MODE = 2 $\quad \pi < \theta_i < 2\pi$

<u>Output</u>:

THTASR	r.s.l. separation angle, θ_s^r

CALLED FROM:	VCF

SUBROUTINE RVTX

PURPOSE:	To calculate the position and strength of the r.s.l. vortices
METHOD:	Subroutine RSEP has determined the r.s.l. separation angle. The strength of the vortex that would be born is $\Gamma = -\Delta t\, u_o^{\circ 2}/2$. If this value of Γ does not exceed .1 it is stored in arrays. The next time cycle (t_{k+1}), a new θ_s^r is calculated, and if the new separation angle exceeds the previous one then the previous point vortex is aborted. However, if the new θ_s^r is less than the previous value of θ_s^r, the previous point vortex is lumped together with the one now in question. After five time cycles ($t_k < t < t_{k+5}$), or when the lumped vortex strength becomes .1 or greater, the lumped vortex is placed into the outer flow. The angle at which it is placed is the average of the five previous separation angles. See part 1 for equations.
USE:	CALL RVTX (X, Y, GMA, THTASR, THTASM, THTAS, UZZ, UZZSQ, GAMA, MODE, KRN, N, NF, IFLAG, BMATL, NDIM, TK)

<u>Input</u>:

AK	$a(t_k)$
DELT	Δt_k
THETAS	θ_s
THTASR	θ_s^r
THTASM	θ_s^r at the time previous to r.s.l. vortex birth
THTAS	Storage array for θ_s^r values
UZZ	Storage array for u_o° values
UZZSQ	Storage array for $u_o^{\circ 2}$ values
GAMA	Storage array for Γ values
MODE	MODE = 1 $0 < \theta < \pi$ MODE = 2 $\pi < \theta < 2\pi$
N	An integer count of the number of time cycles completed since the previous r.s.l. vortex was placed in the outer flow.

	NF	Lump parameter, usually NF = 5. When N = NF a lumped point vortex is placed in the outer flow
	NDIM	Dimension of vortex arrays
	TK	An array of vortex diffusion times
	<u>Output</u>:	
	GMATL	Lumped vortex strength
	IFLAG	If IFLAG(N) = 0, the point vortex aborted If IFLAG(N) = 1, the point vortex is lumped
	X, Y	Coordinates of r.s.l. vortex born
	GMA	Strength of r.s.l. vortex born
	SMALLM	m_{rk}
	KRN	The KRNth element of the r.s.l. vortex arrays
CALLED FROM:	VCF	
SUBPROGRAMS CALLED:	FREEVTX, POTFLOW	

FUNCTION RZRO

PURPOSE: To calculate the nondimensional body radius r_o as a function of nondimensional distance $\hat{z}$ along the body axis.

METHOD: The dimensional quantities are multiplied by characteristic dimensions.

USE: Function reference to

RZRO (ZHAT, L, RW)

<u>Input</u>:

ZHAT	$\hat{z}$
L	ℓ
RW	$d/2$

<u>Output</u>:

RZRO	r_o

CALLED FROM: VCF

SUBPROGRAMS CALLED: RZERO

FUNCTION UAPRX1

PURPOSE: To calculate a first approximation to $u(\theta_{i+1}, t_k, \bar{r}_j)$

METHOD: $u(\theta_{i+1}, t_k, \bar{r}_j) = u(\theta_{i+1}, t_{k-1}, \bar{r}_j) + u(\theta_i, t_k, \bar{r}_j) - u(\theta_i, t_{k-1}, \bar{r}_j$

See reference 1, for further explanation.

USE: Function reference to

UAPRX1 (L, N, IDIM, JDIM, U)

<u>Input</u>:

L, N	The b.l. grid point $(\theta_i, \bar{r}_j)$
IDIM, JDIM	b.l. grid dimension $(\theta_i, \bar{r}_j)$ $i = 1,2, \ldots,$ IDIM $j = 1,2, \ldots,$ JDIM
U	$u(\theta_i, t_{k-1}, \bar{r}_j)$

<u>Output</u>:

U	First approximation to $u(\theta_{i+1}, t_k, \bar{r}_j)$

CALLED FROM: BLBOX, DN

REMARKS: Alternate ENTRY points to UAPRX1 are US, UL, and UM

SUBROUTINE VEL

PURPOSE:	To calculate the velocity of the point vortices	
METHOD:	Each point vortex has a velocity due to the uniform flow around a circular cylinder, plus the velocity induced by all of the other point vortices.	
USE:	CALL VEL (X, Y, XDT, YDT, NDIM, KI, KF, IA)	
	Input:	
	X, Y	Input arrays of vortex locations
	NDIM	Dimension of the vortex arrays
	KI, KF	KF - KI + 1 is the number of vortices in an array
	IA	Vortex field point index (see FREEVTX)
	Output:	
	XDT, YDT	Output arrays of vortex velocities
CALLED FROM:	VCF, RVTX	
SUBPROGRAMS CALLED:	FREEVTX, POTFLOW	

SUBROUTINE VM

PURPOSE:	To calculate the coordinates of the point vortices at time t_{k+1}.
METHOD:	Each point vortex travels $\tilde{u}^{\circ}\ \Delta t_k$ in the x direction, and $\tilde{v}^{\circ}\ \Delta t_k$ in the y direction.
USE:	CALL VM (X, Y, XDT, YDT, NDIM, KI, KF, IA)

<u>Input</u>:

X, Y	Vortex locations at t_k
XDT, YDT	Input arrays of vortex velocities
KI, KF	KF - KI + 1 is the number of vortices in an array
IA	Vortex field point index (see FREEVTX)

<u>Output</u>:

R	Distance from cylinder to vortex
XTEMP, YTEMP	Temporary storage of the input vortex locations
X, Y	Vortex locations at t_{k+1}

CALLED FROM:	VCF
SUBPROGRAMS CALLED:	VMFIX
REMARKS:	VMFIX is called only if $R < a(t_k)$

SUBROUTINE VMFIX

PURPOSE:	To calculate a corrected point vortex location.
METHOD:	The Δt_k approximation involved in the motion of the vortices allows for the possibility that a point vortex may cross the cylinder boundary. Since this is physically impossible VMFIX appends the vortex motion and places the point vortex outside of the cylinder. VMFIX calculates the tangent to the cylinder at the point where a vortex has crossed the cylinder boundary. Appended $\tilde{u}°$, $\tilde{v}°$ are returned to VM such that the point vortex will travel along the tangent line.
USE:	CALL VMFIX (X, Y, XDOT, YDOT)

<u>Input</u>:

X, Y	Vortex location inside of cylinder
XDOT, YDOT	Original $\tilde{u}°$, $\tilde{v}°$
DEF	Some small parameter (.001), applied to the transformations to insure that the appended velocities will result in a new vortex location outside of the cylinder.

<u>Output</u>:

L	Quadrant in which the point vortex is located
XDOT, YDOT	Appended $\tilde{u}°$, $\tilde{v}°$

CALLED FROM:	VCF
REMARKS:	VM calls VMFIX if $R < a(t_k)$(see VM), upon returning to VM the vortex location is recalculated starting with the original vortex position.

SUBROUTINE WRIT

PURPOSE:	Print, punch, vortex locations, strengths, and velocities.
METHOD:	Tape 6 = Output, Tape 7 = Punch
USE:	CALL WRIT (X, Y, XDOT, YDOT, GAMMA, XB, YB, GMAB, KIT, KFT, KIB, KFB, K, NDIM, IW)

<u>Input</u>:

X, Y	Input array of t.b.l. or t.r.s.l. vortex locations
XDOT, YDOT	Input array of t.b.l. or t.r.s.l. vortex velocities
GAMMA	Input array of t.b.l. or t.r.s.l. vortex strengths
XB, YB	Input array of b.b.l. or b.r.s.l. vortex locations
XDOTB, YDOTB	Input array of b.b.l. or b.r.s.l. vortex velocities
GMAB	Input array of b.b.l. or b.r.s.l. vortex strengths
KIT, KFT	KFT - KIT + 1 = number of t.b.l. or t.r.s.l. vortices to be output
KIB, KFB	KFB - KIB + 1 = number of b.b.l. or b.r.s.l. vortices to be output

IW	IW = 6 Print
	IW = 7 Punch
	IW = 9 Write on TAPE 9

CALLED FROM: VCF

APPENDIX Ia

PROGRAM ADYNF AND SUBPROGRAM DESCRIPTIONS

This appendix contains a brief outline of the purpose, method, and use of program ADYNF and subroutine FIT. Subroutine RZRO is described in APPENDIX I. Subroutines SPLINEB and CADRE are not described in this APPENDIX but are listed in APPENDIX II.

PURPOSE: To calculate C_N and $C_{M,\lambda}$

METHOD: Values of $C_D(\hat{z})$ and $\hat{z}$ output from PROGRAM VCF are input on cards to PROGRAM ADYNF. The normal force and moment are given by

$$C_N = (4f/\pi) \int_0^1 c_n \, d\hat{z} \text{ , and}$$

$$C_{M,\lambda} = - (4f/\pi) \int_0^1 [\hat{z} + \frac{r_o}{4f^2} \frac{dr_o}{d\hat{z}}] \, c_n \, d\hat{z} + \lambda \, C_N$$

respectively. Approximations to thse integrals are evaluated by first obtaining a least squares cubic spline approximation to the integrand function and then by integrating the approximate function. The integration interval is divided into sections, and a cubic polynomial approximates the integrand in each section. The endpoints of the sections are called "knots" the approximating cubics being continuous up to the third derivative at the "knots."

USE: Input:

INFO	Data Identification
AATACK	Angle of attack
LAMBDA	Moment arm coefficient
LENGTH	Dimensional body length
F	Fineness Ratio
AW	Characteristic length, $\tilde{a}$
RW	Maximum Radius, d/2
NOKNOT	Number of knots used to segment the interval [0,1]
LX	The number of $C_D(\hat{z})$ values
X	Array of $\hat{z}$ values

PORCD	Array of $C_D(\hat{z})$ values
XI	Array of knot positions, the length of the array equals NOKNOT

<u>Output</u>:

ITER	Number of sweeps through opt,(reference 4)
XI	Knot values after optimization
COEFL	Cubic coefficients
ERRL2	Least square error, spline approximation
ERRL1	Average error, spline approximation
ERRL99	Maximum error, spline approximation
XX	$\hat{z}$
FCTL	Spline approximation to integrand at $\hat{z}$
PRINT	Scaled error, spline approximation
CN	Normal force coefficient C_N
CNI	Approximation to the integral $\int_0^1 c_n \, d\hat{z}$
ERROR	Computed absolute error in CNI (See Appendix II - CADRE)
IFLAG	An integer between 1 and 5 indicating what difficulties were met with in obtaining an approximation to CNI (See Appendix II - CADRE)
CM	Moment coefficient, $C_{M,\lambda}$
CMI	Approximation to the integral $\int_0^1 [\hat{z} + \frac{r_o}{4f^2} \frac{dr_o}{d\hat{z}}] \, c_n \, d\hat{z}$

SUBPROGRAMS CALLED: SPLINEB, CADRE, FIT, RZRO

REMARKS: VCF punched output $\hat{z}$, $C_D(\hat{z})$ is input to ADYNF. The first value of $\hat{z}$ output by VCF is $\hat{z} = .03$, and the final value of $\hat{z}$ is usually slightly less than 1.0. It is known that at $\hat{z} = 0$, $C_D = 0$, and by extrapolating the given data to $\hat{z} = 1.0$, two additional data points are input to ADYNF. See the ADYNF PROGRAM INPUT DATA section, and APPENDIX IV - ADYNF SAMPLE INPUT, for further explanation of the input data.

To evaluate the C_N and $C_{M,\lambda}$ integrals, the integrand functions are approximated using cubic splines. This smooths the data, and allows for the integrand to be approximated at any $\hat{z}$ which permits the use of the integration routine CADRE. CADRE has been proven to be a very successful numerical integration scheme and errors in the calculation of C_N, $C_{M,\lambda}$ are believed to be less than 2%. Subprograms RZRO and RZERO are called by ADYNF. For listings and use of CADRE and SPLINEB see Appendix II.

FUNCTION FIT

PURPOSE: To determine the integrand values $[c_n]$ or

$$[\hat{z} + \frac{r_o}{4f^2} \frac{dr_o}{d\hat{z}}] c_n$$ for arbitrary values of $\hat{z}$, $0 < \hat{z} < 1$.

METHOD: The cubic coefficients have been supplied by SPLINEB. The set of coefficients used to approximate the integrand depends upon knot locations and $\hat{z}$.

USE: Function reference to FIT (X)

Input:

X	$\hat{z}$
XI	Knot locations
COEFL	Cubic coefficients

Output:

FIT	Cubic spline approximation to the integrand function

CALLED FROM: ADYNF

REMARKS: For further information on spline approximations see, references 3 and 4 and SPLINE listing in Appendix II.

APPENDIX Ib

SUBROUTINE IC

This appendix contains a listing of SUBROUTINE IC and subprograms referenced by IC. After the first program cycle is completed SUBROUTINE IC and its subprograms are no longer required, and can be replaced by a dummy SUBROUTINE IC. The coding for the dummy program follows the original SUBROUTINE IC listing.

Description of Parameters

Variable or FUNCTION Subprogram	Description [2]
Q	q
QP	q'
QPP	q''
ETA	η
ZTA001(ETA)	$\zeta'_{oo}(\eta)$
ZTA02A1(ETA)	$\zeta'_{0,2a}(\eta)$
ZTA02B1(ETA)	$\zeta'_{0,2b}(\eta)$
PHIO(ETA)	$\Phi_o(\eta)$
PHI1(ETA)	$\Phi_1(\eta)$
ZTA011(ETA)	$\zeta'_{01}(\eta)$

```
      SUBROUTINE IC(AKTI,U,IDIM,JDIM,MODE)
      DIMENSION U(IDIM,2,JDIM)
      DIMENSION NX(53),DEG(53)
      DIMENSION THTA(53)
      COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL
      COMMON/BLOCK20/DX,DZ,INTX1,NBIG1
      COMMON/BLOCK14/S(53),ST(53),SB(53)
      COMMON/BLOCK30/T,TI,DELT,DELTT,DELTB
      COMMON/BLBOX2/TAU,PT4,NBIG
      COMMON/BLBOX12/ZN(51),ISEP
      COMMON/BLBOX13/KTS,IXTRSET,IXBRSET,UTNBIG(53),UBNBIG(53)
      COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT
      COMMON/BLBOX4/A2,A0,B1,C0,D2,D0,E1,E0
      COMMON/BLBOX5/A4A,A2A,A0A,B3A,B1A,C2A,C0A,D3A,D1A,E4A,E2A,E0A,F0A
      COMMON/BLBOX6/G4A,G2A,G1A,G0A,H3A,H2A,H1A,H0A,I2A,I1A,I0A
      COMMON/BLBOX7/K4A,K2A,K1A,K0A,L3A,L2A,L1A,L0A
      COMMON/BLBOX8/A4B,A2B,A0B,B3B,B1B,C2B,C0B,D3B,D1B,E4B,E2B,E0B,F0B
      COMMON/BLBOX9/G4B,G2B,G1B,G0B,H3B,H2B,H1B,H0B,I2B,I1B,I0B
      COMMON/BLBOX10/K4B,K2B,K1B,K0B,L3B,L2B,L1B,L0B
      REAL IP,KP,LP
      REAL I2,I2A,I2B,I1,I1A,I1B,I0,I0A,I0B
      REAL K4,K4A,K4B,K2,K2A,K2B,K1,K1A,K1B,K0,K0A,K0B
      REAL L3,L3A,L3B,L2,L2A,L2B,L1,L1A,L1B,L0,L0A,L0B
      IF(MODE.EQ.2) GO TO 1
C***********************************************************************
C
C      WUNDT DATA
C
C***********************************************************************
      A2=1.
      A0=-.5
      B1=1.5
      C0=.5
      D2=3.0+4./(3.*PI)
      D0=-.5+2./(3.*PI)
      E1=2.+2./(3.*PI)
      E0=-2./(3.*SQRT(PI))
      A4A=-1.0/3.0
      A2A=-1.0
      A0A=1.0/4.0
      B3A=-2.0/3.0
      B1A=-2.0
      C2A=-5.0/12.0
      C0A=-7.0/6.0
      D3A=9.0/40.0
      D1A=133.0/240.0
      E4A=9.0*SQRT(3.0)/(5.0*PI)
      E2A=27.0*SQRT(3.0)/(5.0*PI)
      E0A=27.0*SQRT(3.0)/(20.0*PI)
      F0A=8.0*SQRT(2.0)/(15.0*SQRT(PI))
      G4A=-5.0/6.0+2.0/(9.0*PI)
      G2A=-5.0/2.0+2.0/(3.0*PI)
      G1A=-4.0/(3.0*SQRT(PI))
      G0A=3.0/8.0-1.0/(2.0*PI)
      H3A=-13.0/12.0+1.0/(3.0*PI)
      H2A=2.0/(3.0*SQRT(PI))
      H1A=-73.0/24.0+23.0/(18.0*PI)
      H0A=-1.0/(3.0*SQRT(PI))
      I2A=-1.0/3.0+1.0/(9.0*PI)
      I1A=1.0/(3.0*SQRT(PI))
      I0A=-5.0/6.0+4.0/(9.0*PI)
      K4A=-1.0/2.0-(2.0+27.0*SQRT(3.0))/(15.0*PI)-64.0/(135.0*PI**2)
      K2A=3.0*K4A
      K1A=-4.0/SQRT(PI)-16.0/(9.0*PI*SQRT(PI))
      K0A=1.0/8.0-(46.0+81.0*SQRT(3.0))/(60.0*PI)-16.0/(45.0*PI**2)
      L3A=-1.0/2.0-(2.0+27.0*SQRT(3.0))/(30.0*PI)-32.0/(135.0*PI**2)
      L2A=2.0/(3.0*SQRT(PI))
      L1A=-19.0/12.0-(22.0+81.0*SQRT(3.0))/(36.0*PI)-16.0/(27.0*PI**2)
      L0A=(-17.0+8.0*SQRT(2.0))/(15.0*SQRT(PI))-16.0/(15.0*PI*SQRT(PI))
      A4B=1.0/3.0
      A2B=0.0
      A0B=1.0/4.0
      B3B=7.0/12.0
      B1B=-3.0/8.0
      C2B=1.0/3.0
      C0B=-5.0/12.0
      D3B=-9.0/160.0
      D1B=-31.0/320.0
      E4B=-E4A/4.0
      E2B=-E2A/4.0
      E0B=-E0A/4.0
      F0B=F0A
      G4B=1.5+2.0/(3.0*PI)
      G2B=.5+2.0/(3.0*PI)
      G1B=0.0
      G0B=5.0/8.0-1.0/(6.0*PI)
      H3B=7.0/4.0+7.0/(9.0*PI)
      H2B=0.0
      H1B=-1.0/8.0+5.0/(6.0*PI)
      H0B=0.0
      I2B=0.5+2.0/(3.0*PI)
      I1B=0.0
      I0B=-1.0/4.0+2.0/(9.0*PI)
      K4B=7.0/6.0+(24.0+9.0*SQRT(3.0))/(20.0*PI)+32.0/(45.0*PI**2)
      K2B=0.5+(136.0+81.0*SQRT(3.0))/(60.0*PI)+32.0/(15.0*PI**2)
      K1B=0.0
      K0B=3.0/8.0+(56.0+81.0*SQRT(3.0))/(240.0*PI)+8.0/(15.0*PI**2)
      L3B=7.0/12.0+(24.0+9.0*SQRT(3.0))/(40.0*PI)+16.0/(45.0*PI**2)
      L2B=0.0
      L1B=-5.0/24.0+(40.0+27.0*SQRT(3.0))/(48.0*PI)+8.0/(9.0*PI**2)
      L0B=(-2.0+8.0*SQRT(2.0))/(15.0*SQRT(PI))-8.0/(45.0*PI*SQRT(PI))
C******************************************************************
C
C      END WUNDT DATA
C
C******************************************************************
 1    CONTINUE
      DO 20 I=1,INTX1
      IF(MODE.EQ.1) S(I)=ST(I)
      IF(MODE.EQ.2) S(I)=SB(I)
      THTA(I)=S(I)
      DEG(I)=THTA(I)*RTOD
C     ********** BC Z=0
      U(I,1,1)=0.0
      U(I,2,1)=0.0
      DO 20 J=2,NBIG
```

```
C********* B.C. THETA=0.0
      IF(I.NE.1) GO TO 11
      U(I,1,J)=0.0
      GO TO 20
   11 IF(J.NE.NBIG) GO TO 10
      U(I,1,NBIG)=2.0*SIN(THTA(I))
      IF(MODE.EQ.1) UTNBIG(I)=U(I,1,NBIG)
      IF(MODE.EQ.2) UBNBIG(I)=U(I,1,NBIG)
      GO TO 20
   10 CONTINUE
      ETA=ZN(J)/(2.0*SQRT(TI))
      Q=2.0*SIN(THTA(I))
      QP=2.0*COS(THTA(I))
      QPP=-2.0*SIN(THTA(I))
      U(I,1,J)=Q*(ZTA001(ETA)+TI*QP*ZTA011(ETA)+TI**2*(QP**2*ZTA02A1
     1(ETA)+Q*QPP*ZTA02B1(ETA)))
   20 CONTINUE
C........ U AT T=TI GOES TO U(BAR) AT T=TI
      DO 21 I=1,INTX1
      UTNBIG(I)=AKTI*UTNBIG(I)
      UBNBIG(I)=AKTI*UBNBIG(I)
      DO 21 J=1,NBIG
   21 U(I,1,J)=AKTI*U(I,1,J)
      IF(MODE.EQ.2) GO TO 9
      IF(LEVEL.GE.4)CALL RITE(1,IDIM,JDIM,U,ZN,NX,S,DEG,ZHAT,DELS,K)
    9 CONTINUE
      WRITE(1) (UTNBIG(I),(U(I,1,J),J=1,NBIG),I=1,INTX1)
      RETURN
      END
```

```
      SUBROUTINE IC(AKTI,U,IDIM,JDIM,MODE)
      DIMENSION U(IDIM,2,JDIM)
      COMMON/BLBOX13/KTS,IXTRSET,IXBRSET,UTNBIG(53),UBNBIG(53)
      READ(1) (UTNBIG(I),(U(I,1,J),J=1,51),I=1,53)
      RETURN
      END
      FUNCTION PHIO(EATA)                                                PHIO    1
      PHIO=ERF(EATA)-1.0                                                 PHIO    2
      RETURN                                                             PHIO    3
      END                                                                PHIO    4
      FUNCTION PHI1(EATA)
      COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DLTA
      COMMON/BLBOX2/TAU,PT4,NBIG
      PHI1=2.0*EXP(-EATA**2)/SQRT(PI)
      RETURN
      END
```

```
ICTION ZTA001(EATA)                                                   ZTA001 1
 ATA                                                                  ZTA001 2
A001=ERF(E)                                                           ZTA001 3
TURN                                                                  ZTA001 4
ND                                                                    ZTA001 5
```

```
FUNCTION ZTA011(EATA)                                                    ZTA011 1
COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DLTA                                   ZTA011 2
COMMON/BLBOX2/TAU,PT4,NBIG                                               ZTA011 3
COMMON/BLBOX4/A2,A0,B1,C0,D2,D0,E1,E0                                    ZTA011 4
E=EATA                                                                   ZTA011 5
A2=1.0                                                                   ZTA011 6
A0=-0.5                                                                  ZTA011 7
B1=1.5                                                                   ZTA011 8
C0=0.5                                                                   ZTA011 9
D2=3.0+4.0/(3.0*PI)                                                      ZTA01 10
D0=-0.5+2.0/(3.0*PI)                                                     ZTA01 11
E1=2.0+2.0/(3.0*PI)                                                      ZTA01 12
E0=-2.0/(3.0*SQRT(PI))                                                   ZTA01 13
ZTA011=(A2*E*E+A0)*PHI0(E)**2+B1*E*PHI0(E)*PHI1(E)+C0*PHI1(E)**2         ZTA01 14
ZTA011=ZTA011+(D2*E*E+D0)*PHI0(E)+(E1*E+E0)*PHI1(E)                      ZTA01 15
RETURN                                                                   ZTA01 16
END                                                                      ZTA01 17
```

```
      FUNCTION ZTA02A1(EATA)
      COMMON/BLBOX5/A4A,A2A,A0A,B3A,B1A,C2A,C0A,D3A,D1A,E4A,E2A,E0A,F0A
      COMMON/BLBOX6/G4A,G2A,G1A,G0A,H3A,H2A,H1A,H0A,I2A,I1A,I0A
      COMMON/BLBOX7/K4A,K2A,K1A,K0A,L3A,L2A,L1A,L0A
      REAL IP,KP,LP
      REAL I2,I2A,I2B,I1,I1A,I1B,I0,I0A,I0B
      REAL K4,K4A,K4B,K2,K2A,K2B,K1,K1A,K1B,K0,K0A,K0B
      REAL L3,L3A,L3B,L2,L2A,L2B,L1,L1A,L1B,L0,L0A,L0B
      E=EATA
      P0=PHI0(E)
      P02=PHI0(E)**2
      P03=PHI0(E)**3
      P1=PHI1(E)
      P12=PHI1(E)**2
      P13=PHI1(E)**3
      ET2=E**2
      ET3=E**3
      ET4=E**4
      A4=A4A
      A2=A2A
      A0=A0A
      B3=B3A
      B1=B1A
      C2=C2A
      C0=C0A
      D3=D3A
      D1=D1A
      E4=E4A
      E2=E2A
      E0=E0A
      F0=F0A
      G4=G4A
      G2=G2A
      G1=G1A
      G0=G0A
      H3=H3A
      H2=H2A
      H1=H1A
      H0=H0A
      I2=I2A
      I1=I1A
      I0=I0A
      K4=K4A
      K2=K2A
      K1=K1A
      K0=K0A
      L3=L3A
      L2=L2A
      L1=L1A
      L0=L0A
      AP=(A4*ET4+A2*ET2+A0)*P03
      BP=(B3*ET3+B1*E)*P02*P1
      CP=(C2*ET2+C0)*P0*P12
      DP=(D3*ET3+D1*E)*P13
      SQRT2E=SQRT(2.0)*E
      SQRT3E=SQRT(3.0)*E
      EP=(E4*ET4+E2*ET2+E0)*PHI0(SQRT3E)
      FP=F0*PHI0(SQRT2E)*P1
      GP= (G4*ET4+G2*ET2+G1*E+G0)*P02
      HP= (H3*ET3+H2*ET2+H1*E+H0)*P0*P1
      IP= (I2*ET2+I1*E+I0)*P12
      KP= (K4*ET4+K2*ET2+K1*E+K0)*P0
      LP= (L3*ET3+L2*ET2+L1*E+L0)*P1
      ZTA02A1=AP+BP+CP+DP+EP+FP+GP+HP+IP+KP+LP
      RETURN
      END
      FUNCTION ZTA02B1(EATA)
      COMMON/BLBOX8/A4B,A2B,A0B,B3B,B1B,C2B,C0B,D3B,D1B,E4B,E2B,E0B,F0B
      COMMON/BLBOX9/G4B,G2B,G1B,G0B,H3B,H2B,H1B,H0B,I2B,I1B,I0B
~~      COMMON/BLBOX10/K4B,K2B,K1B,K0B,L3B,L2B,L1B,L0B~~
      REAL IP,KP,LP
      REAL I2,I2A,I2B,I1,I1A,I1B,I0,I0A,I0B
      REAL K4,K4A,K4B,K2,K2A,K2B,K1,K1A,K1B,K0,K0A,K0B
      REAL L3,L3A,L3B,L2,L2A,L2B,L1,L1A,L1B,L0,L0A,L0B
      E=EATA
~~      P0=PHI0(E)~~
      P02=PHI0(E)**2
      P03=PHI0(E)**3
~~      P1=PHI1(E)~~
      P12=PHI1(E)**2
      P13=PHI1(E)**3
~~      ET2=E**2~~
      ET3=E**3
      ET4=E**4
      A4=A4B
      A2=A2B
      A0=A0B
~~      B3=B3B~~
      B1=B1B
      C2=C2B
      C0=C0B
      D3=D3B
      D1=D1B
~~      E4=E4B~~
      E2=E2B
      E0=E0B
      F0=F0B
      G4=G4B
      G2=G2B
~~      G1=G1B~~
      G0=G0B
      H3=H3B
      H2=H2B
      H1=H1B
      H0=H0B
~~      I2=I2B~~
      I1=I1B
      I0=I0B
      K4=K4B
      K2=K2B
      K1=K1B
~~      K0=K0B~~
      L3=L3B
      L2=L2B
      L1=L1B
      L0=L0B
      AP=(A4*ET4+A2*ET2+A0)*P03
~~      BP=(B3*ET3+B1*E)*P02*P1~~
      CP=(C2*ET2+C0)*P0*P12
      DP=(D3*ET3+D1*E)*P13
      SQRT2E=SQRT(2.0)*E
      SQRT3E=SQRT(3.0)*E
      EP=(E4*ET4+E2*ET2+E0)*PHI0(SQRT3E)
~~      FP=F0*PHI0(SQRT2E)*P1~~
```

```
GP= (G4*ET4+G2*ET2+G1*E+G0)*P02
HP= (H3*ET3+H2*ET2+H1*E+H0)*P0*P1
IP= (I2*ET2+I1*E+I0)*P12
KP= (K4*ET4+K2*ET2+K1*E+K0)*P0
LP= (L3*ET3+L2*ET2+L1*E+L0)*P1
ZTA02B1=AP+BP+CP+DP+EP+FP+GP+HP+IP+KP+LP
RETURN
END
```

APPENDIX II

CANNED SUBROUTINE LISTINGS - CADRE

- SECOND
- SPLINE, ARITH1
- TRID

```
      FUNCTION CADRE(F,A,B,AERR,RERR,LEVEL,ERROR,IFLAG)
C  THIS FUNCTION RETURNS AN ESTIMATE *CADRE* FOR THE NUMBER
C     INT = INTEGRAL OF *F*(X) FROM *A* TO *B*
C  WHICH HOPEFULLY SATISFIES
C     ABS(INT - *CADRE*)  .LE.  AMAX1(*AERR*, *RERR* TIMES ABS(INT)).
C     THE PROGRAM USES CAUTIOUS ADAPTIVE ROMBERG EXTRAPOLATION.
C  IN THIS SCHEME, THE INTEGRAL IS CALCULATED AS THE SUM OF INTEGRALS
C  OVER SUITABLY SMALL SUBINTERVALS. ON EACH SUBINTERVAL, AN ESTIMATE
C  *VINT*, WITH ESTIMATED ABSOLUTE ERROR *ERRER*, IS FOUND BY CAUTIOUS
C  ROMBERG EXTRAPOLATION. IF *ERRER* IS SMALL ENOUGH, *VINT* IS ACCEPT
C  ED AND ADDED TO *CADRE*, AND *ERRER* IS ADDED TO *ERROR*. OTHERWISE
C  THE SUBINTERVAL IS HALVED, AND EACH HALF IS CONSIDERED SEPARATELY,
C  INFORMATION ABOUT THE OTHER HALF BEING TEMPORARILY STACKED.
C
C                    *****   INPUT   *****
C
C  F        THE NAME OF A SINGLE-ARGUMENT REAL FUNCTION SUBPROGRAM
C           THIS NAME MUST APPEAR IN THE CALLING PROGRAM IN AN
C           EXTERNAL STATEMENT.
C  A,B      THE TWO ENDPOINTS OF THE INTERVAL OF INTEGRATION
C  AERR
C  RERR     DESIRED ABSOLUTE AND RELATIVE ERROR IN THE ANSWER
C  LEVEL    AN INTEGER INDICATING DESIRED LEVEL OF PRINTOUT
C          .LE. 1,  NO PRINTOUT,
C            =  2,  SUCCESS OR FAILURE MESSAGE, AND LIST OF SINGULAR-
C                   ITIES ENCOUNTERED (IF ANY).
C            =  3,  IN ADDITION, ALL SUBINTERVALS CONSIDERED ARE LISTED
C                   TOGETHER WITH THE KIND OF REGULAR BEHAVIOUR FOUND
C                   (IF ANY),
C            =  4,  IN ADDITION, ALL RATIOS CONSIDERED ARE LISTED AS IS
C                   INFO ON WHICH DECISION PROCEDURE IS BASED,
C          .GE. 5,  IN ADDITION, ALL T-TABLES ARE LISTED.
C
C                    *****   OUTPUT   *****
C
C  CADRE    ESTIMATE OF THE INTEGRAL, RETURNED VIA THE FUNCTION CALL.
C  ERROR    ESTIMATED BOUND ON THE ABSOLUTE ERROR OF THE NUMBER *CADRE*
C  IFLAG    AN INTEGER BETWEEN 1 AND 5 INDICATING WHAT DIFFICULTIES
C           WERE MET WITH, SPECIFICALLY
C           = 1, ALL IS WELL.
C           = 2, ONE ORE MORE SINGULARITIES WERE SUCCESSFULLY HANDLED.
C           = 3, IN SOME SUBINTERVAL(S), THE ESTIMATE *VINT* WAS ACCEPT
C                ED MERELY BECAUSE *ERRER* WAS SMALL, EVEN THOUGH NO
C                REGULAR BEHAVIOUR COULD BE RECOGNIZED.
C           = 4, FAILURE, OVERFLOW OF STACK *TS* (THIS HAS NEVER HAPPEN
C                ED, - SO FAR),
C           = 5, FAILURE, TOO SMALL A SUBINTERVAL IS REQUIRED, THIS MAY
C                BE DUE TO TOO MUCH NOISE IN THE FUNCTION (RELATIVE TO
C                THE GIVEN ERROR REQUIREMENTS) OR DUE TO A PLAIN ORNERY
C                INTEGRAND.
C  A VERY CAUTIOUS MAN WOULD ACCEPT *CADRE* ONLY IF IFLAG IS 1 OR 2.
C  THE MERELY REASONABLE MAN WOULD KEEP THE FAITH EVEN IF IFLAG IS 3.
C  THE ADVENTUROUS MAN IS QUITE OFTEN RIGHT IN ACCEPTING *CADRE*
C  EVEN IF IFLAG  IS 4 OR 5.
C
C      *****   LIST OF MAJOR VARIABLES   *****
C
C  CUREST   BEST ESTIMATE SO FAR FOR
C               INT - (INTEGRAL OVER CURRENTLY CONSIDERED SUBINTERVAL).
C  FNSIZE   MAXIMUM AVERAGE FUNCTION SIZE SO FAR ENCOUNTERED,
C   ERRR     RELATIVE ERROR REQUIREMENT USED; DERIVED FROM INPUT *RERR*
C            AND CHOSEN TO LIE BETWEEN .1 AND 10 TIMES *TOLMCH*.
C   ERRA     = ABS(*AERR*)
C   STAGE    (MORE OR LESS) EQUAL TO 2 TO THE -(*ISTAGE*)
C   THESE FIVE QUANTITIES ARE USED IN THE DETERMINATION OF THE LOCAL
C   ERROR REQUIREMENT.
C
C   STEPMN   MINIMUM SUBINTERVAL LENGTH PERMITTED.
C   TS       STACK OF VALUES OF F(X) SO FAR COMPUTED BUT NOT YET
C            SUCCESSFULLY USED.
C   ISTAGE   AN INTEGER INDICATING THE HEIGHT OF THE STACK OF INTERVALS
C            YET TO BE PROCESSED.
C
C      *****   LIST OF PARAMETERS   *****
C
C   TOLMCH   DEPENDS ON THE LENGTH OF FLOATING POINT MANTISSA; SHOULD BE
C            ABOUT 1.E-7 FOR 27 BINARY BIT MANTISSA AND
C            ABOUT 1.E-13 FOR 48 BINARY BIT MANTISSA.
C   AITLOW   SHOULD BE SOMEWHAT GREATER THAN 1.
C   H2TOL.
C   AITTOL.
C   JUMPTL   TOLERANCES USED IN THE DECISION PROCESS TO RECOGNIZE
C            H**2 CONVERGENCE, X**ALPHA TYPE CONVERGENCE, OR
C            JUMP-TYPE CONVERGENCE OF THE TRAPEZOID SUMS.
C   MAXTS.
C   MAXTBL.
C   MXSTGE   ARE THE THREE DIFFERENT UPPER LIMITS FOR THE DIMENSION OF
C            THE VARIOUS ARRAYS.
C
C        *****   PROGRAM LAYOUT   *****
C
C            INITIALIZATION.
C   5.6      BEGIN WORK ON NEXT SUBINTERVAL
C    9-14    GET NEXT TRAPEZOID SUM
C   15-19    GET RATIOS. PRELIMINARY DECISION PROCEDURE.
C   20-      ESTIMATE *VINT* ASSUMING SMOOTH INTEGRAND
C   30-      ESTIMATE *VINT* ASSUMING INTEGRAND HAS X**ALPHA TYPE
C            SINGULARITY
C   40-      NO LUCK WITH THIS TRAPEZOID SUM. GET NEXT ONE OR GET OUT.
C   50-      ESTIMATE *VINT* ASSUMING INTEGRAND HAS JUMP
C   60-      ESTIMATE *VINT* ASSUMING INTEGRAND IS STRAIGHT LINE
C   70-      ESTIMATE *VINT* ASSUMING VARIATION IN INTEGRAND
C            IS MOSTLY NOISE.
C   80-      INTEGRATION OVER CURRENT SUBINTERVAL SUCCESSFUL.
C            SET UP NEXT SUBINTERVAL, IF ANY, OR RETURN.
C   90-      INTEGRATION OVER CURRENT SUBINTERVAL NOT SUCCESSFUL.
C            MARK CURRENT SUBINTERVAL FOR SUBDIVISION AND SET UP
C            NEXT SUBINTERVAL.
C   900-     FAILURE.
C
      DIMENSION T(10,10),R(10),AIT(10),DIF(10),RN(4),
     *          TS(2049),IBEGS(30),BEGIN(30),FINIS(30),EST(30)
      REAL LENGTH, JUMPTL
      LOGICAL H2CONV,AITKEN,RIGHT,REGLAR,REGLSV(30)
      DATA TOLMCH,AITLOW,H2TOL,AITTOL,JUMPTL,MAXTS,MAXTBL,MXSTGE
     *  / 2.E-13, 1.1  , .15 ,  .1  , .01  , 2049,  10  ,  30/
      DATA RN/.71420053,.34662815,.843751,.12633046/
      DATA ALG402 /.3010299956639795/
      CADRE = 0.
      ERROR = 0.
      IFLAG = 1
      LENGTH = ABS(B-A)                                                 CADRE122
      IF (LENGTH .EQ. 0.)                RETURN                         CADRE123
      ERRR = AMIN1(.1,AMAX1(ABS(RERR), 10.*TOLMCH))                     CADRE124
      ERRA = ABS(AERR)                                                  CADRE125
      STEPMN = AMAX1(LENGTH/FLOAT(2**MXSTGE),                           CADRE126
     *          AMAX1(LENGTH,ABS(A),ABS(B))*TOLMCH)                     CADRE127
      STAGE = .5                                                        CADRE128
      ISTAGE = 1                                                        CADRE129
      CUREST = 0.                                                       CADRE130
      FNSIZE = 0.                                                       CADRE131
      PREVER = 0.                                                       CADRE132
      REGLAR = .FALSE.                                                  CADRE133
C                                                                       CADRE134
C  THE GIVEN INTERVAL OF INTEGRATION IS THE FIRST INTERVAL CONSIDERED.  CADRE135
      BEG = A                                                           CADRE136
      FBEG = F(BEG)/2.                                                  CADRE137
      TS(1) = FBEG                                                      CADRE138
      IBEG = 1                                                          CADRE139
      END = B                                                           CADRE140
      FEND = F(END)/2.                                                  CADRE141
      TS(2) = FEND                                                      CADRE142
      IEND = 2                                                          CADRE143
C                                                                       CADRE144
    5 RIGHT = .FALSE.                                                   CADRE145
C                                                                       CADRE146
C  INVESTIGATION OF A PARTICULAR SUBINTERVAL BEGINS AT THIS POINT.      CADRE147
C           *****    MAJOR VARIABLES    *****                           CADRE148
C  BEG,                                                                 CADRE149
C  END      ENDPOINTS OF THE CURRENT INTERVAL                           CADRE150
C  FBEG,                                                                CADRE151
C  FEND     ONE HALF THE VALUE OF F(X) AT THE ENDPOINTS                 CADRE152
C  STEP     SIGNED LENGTH OF CURRENT SUBINTERVAL                        CADRE153
C  ISTAGE   HEIGHT OF CURRENT SUBINTERVAL IN STACK OF SUBINTERVALS      CADRE154
C           YET TO BE DONE                                              CADRE155
C  RIGHT    A LOGICAL VARIABLE INDICATING WHETHER CURRENT SUBINTERVAL   CADRE156
C           IS RIGHT HALF OF PREVIOUS SUBINTERVAL. NEEDED IN 80FF AND   CADRE157
C           90FF TO DECIDE WHAT INTERVAL TO LOOK AT NEXT.               CADRE158
C  TS(I), I=IBEG,...,IEND, CONTAINS THE FUNCTION VALUES FOR THIS        CADRE159
C           SUBINTERVAL SO FAR COMPUTED. SPECIFICALLY,                  CADRE160
C             TS(I) = F(BEG + (I-IBEG)/(IEND-IBEG)*STEP), ALL I         CADRE161
C           EXCEPT THAT TS(IBEG) = FBEG, TS(IEND) = FEND                CADRE162
C  REGLAR   A LOGICAL VARIABLE INDICATING WHETHER OR NOT THE CURRENT    CADRE163
C           SUBINTERVAL IS REGULAR (SEE NOTES)                          CADRE164
C  H2CONV   A LOGICAL VARIABLE INDICATING WHETHER H**2 CONVERGENCE OF   CADRE165
C           THE TRAPEZOID SUMS FOR THIS INTERVAL IS RECOGNIZED,         CADRE166
C  AITKEN   A LOGICAL VARIABLE INDICATING WHETHER CONVERGENCE OF RATIOS CADRE167
C           FOR THIS SUBINTERVAL IS RECOGNIZED                          CADRE168
C  T        CONTAINS THE FIRST *L* ROWS OF THE ROMBERG T-TABLE FOR THIS CADRE169
C           SUBINTERVAL IN ITS LOWER TRIANGULAR PART. SPECIFICALLY,     CADRE170
C             T(I,1) = TRAPEZOID SUM (WITHOUT THE FACTOR *STEP*)        CADRE171
C                      ON 2**(I-1) + 1 EQUISPACED POINTS, I=1,...,L,    CADRE172
C             T(I,J+1) = T(I,J) + (T(I,J)-T(I-1,J))/(4**J - 1),         CADRE173
C                      J=2,...,I-1, I=2,...,L.                          CADRE174
C           FURTHER, THE STRICTLY UPPER TRIANGULAR PART OF T CONTAINS   CADRE175
C           THE RATIOS FOR THE VARIOUS COLUMNS OF THE T-TABLE.          CADRE176
C           SPECIFICALLY,                                               CADRE177
C             T(J,I) = (T(I,J)-T(I-1,J))/(T(I+1,J)-T(I,J)),             CADRE178
C                      I=J+1,...,L-1,  J=1,...,L-2.                     CADRE179
C           FINALLY, THE LAST OR L-TH COLUMN CONTAINS                   CADRE180
C             T(J,L) = T(L,J) - T(L-1,J), J=1,...,L-1.                  CADRE181
    6 STEP = END - BEG                                                  CADRE182
```

```
      ASTEP =  ABS(STEP)
      IF (ASTEP .LT. STEPMN)                  GO TO 950
      IF (LEVEL .GE. 3)  WRITE(6,609) BEG,STEP,ISTAGE
  609 FORMAT(10H BEG,STEP ,2E16.8,I5)
      T(1,1) = FBEG + FEND
      TABS = ABS(FBEG) + ABS(FEND)
      L = 1
      N = 1
      H2CONV = .FALSE.
      AITKEN = .FALSE.
                                              GO TO 10
C
    9 IF (LEVEL .GE. 4)  WRITE(6,692) L,T(1,LM1)
   10 LM1 = L
      L = L + 1
C
CALCULATE THE NEXT TRAPEZOID SUM, T(L,1), WHICH IS BASED ON
C  *N2*  + 1 EQUISPACED POINTS. HERE,  N2 = N*2 = 2**(L-1) .
      N2 = N*2
      FN = N2
      ISTEP = (IEND - IBEG)/N
      IF (ISTEP .GT. 1)                       GO TO 12
      II = IEND
      IEND = IEND + N
      IF (IEND .GT. MAXTS)                    GO TO 900
      HOVN = STEP/FN
      III = IEND
      DO 11 I=1,N2,2
      TS(III) = TS(II)
      TS(III-1) = F(END - FLOAT(I)*HOVN)
      III = III-2
   11 II = II-1
      ISTEP = 2
   12 ISTEP2 = IBEG + ISTEP/2
      SUM = 0.
      SUMABS = 0.
      DO 13 I=ISTEP2,IEND,ISTEP
      SUM = SUM + TS(I)
   13 SUMABS = SUMABS + ABS(TS(I))
      T(L,1) = T(L-1,1)/2. + SUM/FN
      TABS = TABS/2. + SUMABS/FN
      ABSI = ASTEP*TABS
      N = N2
C
C  GET PRELIMINARY VALUE FOR *VINT* FROM LAST TRAPEZOID SUM AND
C  UPDATE THE ERROR REQUIREMENT *ERGOAL* FOR THIS SUBINTERVAL.
C  THE ERROR REQUIREMENT IS NOT PRORATED ACCORDING TO THE LENGTH OF
C  THE CURRENT SUBINTERVAL RELATIVE TO THE INTERVAL OF INTEGRATION,
C  BUT ACCORDING TO THE HEIGHT *ISTAGE* OF THE CURRENT SUBINTERVAL
C  IN THE STACK OF SUBINTERVALS YET TO BE DONE.
C  THIS PROCEDURE IS NOT BACKED BY ANY RIGOROUS ARGUMENT, BUT
C  SEEMS TO WORK.
      IT = 1
      VINT = STEP*T(L,1)
      TABTLM = TABS*TOLMCH
      FNSIZE = AMAX1(FNSIZE,ABS(T(L,1)))
      ERGOAL = AMAX1(ASTEP*TOLMCH*FNSIZE,
     *        STAGE*AMAX1(ERRA,ERRR*ABS(CUREST+VINT)))
C
COMPLETE ROW L AND COLUMN L OF *T* ARRAY.
      FEXTRP = 1.
      DO 14 I=1,LM1
      FEXTRP = FEXTRP*4.
      T(I,L) = T(L,I) - T(L-1,I)
   14 T(L,I+1) = T(L,I) + T(I,L)/(FEXTRP-1.)
      ERRER = ASTEP*ABS(T(1,L))
C-----------------------------------------------------------------------
C---  PRELIMINARY DECISION PROCEDURE  ----------------------------------
C
C  IF L = 2 AND T(2,1) = T(1,1), GO TO 60 TO FOLLOW UP THE IMPRESSION
C  THAT INTEGRAND IS STRAIGHT LINE.
      IF (L .GT. 2)                  GO TO 15
      IF (ABS(T(1,2)) .LE. TABTLM)   GO TO 60
                                     GO TO 10
C
CALCULATE NEXT RATIOS FOR COLUMNS 1,...,L-2 OF T-TABLE
C  RATIO IS SET TO ZERO IF DIFFERENCE IN LAST TWO ENTRIES OF
C  COLUMN IS ABOUT ZERO.
   15 DO 16 I=2,LM1
      DIFF = 0.
      IF (ABS(T(I-1,L)) .GT. TABTLM) DIFF = T(I-1,LM1)/T(I-1,L)
   16 T(I-1,LM1) = DIFF
C
C  T(1,LM1) IS THE RATIO DERIVED FROM LAST THREE TRAPEZOID SUMS, I.E.,
C     T(1,LM1) = (T(L-1,1)-T(L-2,1))/(T(L,1)-T(L-1,1)) .
C  IF THIS RATIO IS ABOUT 4, GO TO 20 FOR ROMBERG EXTRAPOLATION.
C  IF THIS RATIO IS ZERO, I.E., IF LAST TWO TRAPEZOID SUMS ARE ABOUT
C     EQUAL, GO TO 18 FOR POSSIBLE NOISE CHECK.
C  IF THIS RATIO IS ABOUT 2 IN ABSOLUTE VALUE, GO TO 50 WITH THE
C     BELIEF THAT INTEGRAND HAS JUMP DISCONTINUITY.
C  IF THIS RATIO IS, AT LEAST, ABOUT EQUAL TO THE PREVIOUS RATIO, THEN
C     THE INTEGRAND MAY WELL HAVE A NICE INTEGRABLE SINGULARITY.
C     GO TO 30 TO FOLLOW UP THIS HUNCH.
      IF (ABS(4.-T(1,LM1)) .LE. H2TOL) GO TO 20
      IF (T(1,LM1) .EQ. 0.)          GO TO 18
      IF (ABS(2.-ABS(T(1,LM1))) .LT. JUMPTL) GO TO 50
      IF (L .EQ. 3)                  GO TO 9
      H2CONV = .FALSE.
      IF (ABS((T(1,LM1)-T(1,L-2))/T(1,LM1)) .LE. AITTOL)
     *                               GO TO 30
C  AT THIS POINT, NO REGULAR BEHAVIOUR WAS DETECTED.
C  IF CURRENT SUBINTERVAL IS NOT REGULAR AND ONLY FOUR TRAPEZOID SUMS
C  WERE COMPUTED SO FAR, TRY ONE MORE TRAPEZOID SUM.
C  IF , AT LEAST, LAST TWO TRAPEZOID SUMS ARE ABOUT EQUAL, THEN
C  FAILURE TO RECOGNIZE REGULAR BEHAVIOUR MAY WELL BE DUE TO NOISE.
C  GO TO 70 TO CHECK THIS OUT.
C  OTHERWISE, GO TO 90 FOR FURTHER SUBDIVISION.
C
   17 IF (REGLAR)                    GO TO 18
      IF (L .EQ. 4)                  GO TO 9
   18 IF (ERRER .LE. ERGOAL)         GO TO 70
      IF (LEVEL .GE. 4)  WRITE (6,692) L,T(1,LM1)
                                     GO TO 91
C-----------------------------------------------------------------------
CAUTIOUS ROMBERG EXTRAPOLATION  ----------------------------------------
C
C  THE CURRENT, OR L-TH, ROW OF THE ROMBERG T-TABLE HAS L ENTRIES.
C  FOR J=1,...,L-2, THE ESTIMATE
C              STEP*T(L,J+1)
C  IS BELIEVED TO HAVE ITS ERROR BOUNDED BY
C    ABS(STEP*(T(L,J)-T(L-1,J))/(4**J - 1))
C  IF THE LAST RATIO
```

```
C        T(J,LM1) = (T(L-1,J)-T(L-2,J))/(T(L,J)-T(L-1,J))               CADRE305
C  FOR COLUMN J OF THE T-TABLE IS ABOUT 4**J.                           CADRE306
C  THE FOLLOWING IS A SLIGHTLY RELAXED EXPRESSION OF THIS BELIEF.       CADRE307
   20 IF (LEVEL .GE. 4) WRITE (6,619) L,T(1,LM1)                        CADRE308
  619 FORMAT(I5,E16.8,5X6HH2CONV)                                       CADRE309
      IF (H2CONV)                               GO TO 21                CADRE310
      AITKEN = .FALSE.                                                  CADRE311
      H2CONV = .TRUE.                                                   CADRE312
      IF (LEVEL .GE. 3)  WRITE (6,620) L                                CADRE313
  620 FORMAT(22H H2 CONVERGENCE AT ROW,I3)                              CADRE314
   21 FEXTRP = 4.                                                       CADRE315
   22 IT = IT + 1                                                       CADRE316
      VINT = STEP*T(L,IT)                                               CADRE317
      ERRER = ABS(STEP/(FEXTRP-1.)*T(IT-1,L))                           CADRE318
      IF (ERRER .LE. ERGOAL)                    GO TO 80                CADRE319
      IF (IT .EQ. LM1)                          GO TO 40                CADRE320
      IF (T(IT,LM1) .EQ. 0.)                    GO TO 22                CADRE321
      IF (T(IT,LM1) .LE. FEXTRP)                GO TO 40                CADRE322
      IF (ABS(T(IT,LM1)/4.-FEXTRP)/FEXTRP .LT. AITTOL)                  CADRE323
     *        FEXTRP = FEXTRP*4.                                        CADRE324
                                                GO TO 22                CADRE325
C-----------------------------------------------------------------------CADRE326
C--- INTEGRAND MAY HAVE X**ALPHA TYPE SINGULARITY-----------------------CADRE327
C  RESULTING IN A RATIO OF *SING*  =  2**(ALPHA + 1)                    CADRE328
   30 IF (LEVEL .GE. 4) WRITE (6,629) L,T(1,LM1)                        CADRE329
  629 FORMAT(I5,E16.8,5X6HAITKEN)                                       CADRE330
      IF (T(1,LM1) .LT. AITLOW)                 GO TO 91                CADRE331
      IF (AITKEN)                               GO TO 31                CADRE332
      H2CONV = .FALSE.                                                  CADRE333
      AITKEN = .TRUE.                                                   CADRE334
      IF (LEVEL .GE. 3) WRITE (6,630) L                                 CADRE335
  630 FORMAT(14H AITKEN AT ROW,I3)                                      CADRE336
   31 FEXTRP = T(L-2,LM1)                                               CADRE337
      IF (FEXTRP .GT. 4.5)                      GO TO 21                CADRE338
      IF (FEXTRP .LT. AITLOW)                   GO TO 91                CADRE339
      IF (ABS(FEXTRP-T(L-3,LM1))/T(1,LM1) .GT. H2TOL)                   CADRE340
     *                                          GO TO 91                CADRE341
      IF (LEVEL .GE. 3) WRITE (6,631) FEXTRP                            CADRE342
  631 FORMAT(6H RATIO,F12.8)                                            CADRE343
      SING = FEXTRP                                                     CADRE344
      FEXTM1 = FEXTRP - 1.                                              CADRE345
      DO 32 I=2,L                                                       CADRE346
      AIT(I) = T(I,1) + (T(I,1)-T(I-1,1))/FEXTM1                        CADRE347
      R(I) = T(1,I-1)                                                   CADRE348
   32 DIF(I) = AIT(I) - AIT(I-1)                                        CADRE349
      IT = 2                                                            CADRE350
   33 VINT = STEP*AIT(L)                                                CADRE351
      IF (LEVEL .LT. 5) GO TO 333                                       CADRE352
      WRITE (6,632) (R(I+1),I=IT,LM1)                                   CADRE353
      WRITE (6,632) (AIT(I),I=IT,L)                                     CADRE354
      WRITE (6,632) (DIF(I+1),I=IT,LM1)                                 CADRE355
  632 FORMAT(1X,8E15.8)                                                 CADRE356
  333 ERRER = ERRER/FEXTM1                                              CADRE357
      IF (ERRER .GT. ERGOAL)                    GO TO 34                CADRE358
      ALPHA = ALOG10(SING)/ALG402 - 1.                                  CADRE359
      IF (LEVEL .GE. 2) WRITE (6,633) ALPHA,BEG,END                     CADRE360
  633 FORMAT(11X42HINTEGRAND SHOWS SINGULAR BEHAVIOR OF TYPE            CADRE361
     *        4HX**(F4.2,9H) BETWEENE15.8,4H ANDE15.8)                  CADRE362
      IFLAG = MAX0(IFLAG,2)                                             CADRE363
                                                GO TO 80                CADRE364
   34 IT = IT + 1                                                       CADRE365
```

```
      IF (IT .EQ. LM1)                    GO TO 40                       CADRE366
      IF (IT .GT. 3)                      GO TO 35                       CADRE367
      H2NEXT = 4.                                                        CADRE368
      SINGNX = 2.*SING                                                   CADRE369
   35 IF (H2NEXT .LT. SINGNX)             GO TO 36                       CADRE370
      FEXTRP = SINGNX                                                    CADRE371
      SINGNX = 2.*SINGNX                                                 CADRE372
                                          GO TO 37                       CADRE373
   36 FEXTRP = H2NEXT                                                    CADRE374
      H2NEXT = 4.*H2NEXT                                                 CADRE375
   37 DO 38 I=IT,LM1                                                     CADRE376
      R(I+1) = 0.                                                        CADRE377
   38 IF (ABS(DIF(I+1)) .GT. TABTLM) R(I+1) = DIF(I)/DIF(I+1)            CADRE378
      IF (LEVEL .GE. 4) WRITE (6,638) FEXTRP,R(L-1),R(L)                 CADRE379
  638 FORMAT(16H FEXTRP + RATIOS,3E15.8)                                 CADRE380
      H2TFEX = -H2TOL*FEXTRP                                             CADRE381
      IF (R(L) - FEXTRP .LT. H2TFEX)      GO TO 40                       CADRE382
      IF (R(L-1)-FEXTRP .LT. H2TFEX)      GO TO 40                       CADRE383
      ERRER = ASTEP*ABS(DIF(L))                                          CADRE384
      FEXTM1 = FEXTRP - 1.                                               CADRE385
      DO 39 I=IT,L                                                       CADRE386
      AIT(I) = AIT(I) + DIF(I)/FEXTM1                                    CADRE387
   39 DIF(I) = AIT(I) - AIT(I-1)                                         CADRE388
                                          GO TO 33                       CADRE389
C----------------------------------------------------------------------- CADRE390
CURRENT TRAPEZOID SUM AND RESULTING EXTRAPOLATED VALUES DID NOT GIVE     CADRE391
C  A SMALL ENOUGH *ERRER*.                                               CADRE392
C  IF LESS THAN FIVE TRAPEZOID SUMS WERE COMPUTED SO FAR, TRY NEXT       CADRE393
C     TRAPEZOID SUM.                                                     CADRE394
C  OTHERWISE, DECIDE WHETHER TO GO ON OR TO SUBDIVIDE AS FOLLOWS.        CADRE395
C  WITH T(L,IT) GIVING THE CURRENTLY BEST ESTIMATE, GIVE UP ON DEVELOP   CADRE396
C  ING THE T-TABLE FURTHER IF  L .GT. IT+2, I.E., IF EXTRAPOLATION       CADRE397
C  DID NOT GO VERY FAR INTO THE T-TABLE.                                 CADRE398
C  FURTHER, GIVE UP IF REDUCTION IN *ERRER* AT THE CURRENT RATE          CADRE399
C  DOES NOT PREDICT AN *ERRER* LESS THAN *ERGOAL* BY THE TIME            CADRE400
C  *MAXTBL* TRAPEZOID SUMS HAVE BEEN COMPUTED.                           CADRE401
C  ---NOTE---                                                            CADRE402
C  HAVING  PREVER .LT. ERRER  IS AN ALMOST CERTAIN SIGN OF BEGINNING     CADRE403
C  TROUBLE WITH NOISE IN THE FUNCTION VALUES. HENCE,                     CADRE404
C  A WATCH FOR, AND CONTROL OF, NOISE SHOULD BEGIN HERE.                 CADRE405
   40 FEXTRP = AMAX1(PREVER/ERRER,AITLOW)                                CADRE406
      PREVER = ERRER                                                     CADRE407
      IF (L .LT. 5)                       GO TO 10                       CADRE408
      IF (LEVEL .GE. 3) WRITE (6,641) ERRER,ERGOAL,FEXTRP,IT             CADRE409
  641 FORMAT(23H ERRER,ERGOAL,FEXTRP,IT,2E15.8,E14.5,I3)                 CADRE410
      IF (L-IT .GT. 2 .AND. ISTAGE .LT. MXSTGE) GO TO 90                 CADRE411
      IF (ERRER/FEXTRP**(MAXTBL-L) .LT. ERGOAL) GO TO 10                 CADRE412
                                          GO TO 90                       CADRE413
C----------------------------------------------------------------------- CADRE414
C----     INTEGRAND HAS JUMP (SEE NOTES)  ------------------------------- CADRE415
   50 IF (LEVEL .GE. 4) WRITE (6,649) L,T(1,LM1)                         CADRE416
  649 FORMAT(I5,E16.8,5X4HJUMP)                                          CADRE417
      IF (ERRER .GT. ERGOAL)              GO TO 90                       CADRE418
C  NOTE THAT  2*FN = 2**L                                                CADRE419
      DIFF = ABS(T(1,L))*2.*FN                                           CADRE420
      IF (LEVEL .GE. 2) WRITE (6,650) DIFF,BEG,END                       CADRE421
  650 FORMAT(13X36HINTEGRAND SEEMS TO HAVE JUMP OF SIZEE13.6,            CADRE422
     *         8H BETWEENE15.8,4H ANDE15.8)                              CADRE423
                                          GO TO 80                       CADRE424
C----------------------------------------------------------------------- CADRE425
C----  INTEGRAND IS STRAIGHT LINE  ------------------------------------- CADRE426
```

```
C  TEST THIS ASSUMPTION BY COMPARING THE VALUE OF THE INTEGRAND AT          CADRE427
C  FOUR *RANDOMLY CHOSEN* POINTS WITH THE VALUE OF THE STRAIGHT LINE        CADRE428
C  INTERPOLATING THE INTEGRAND AT THE TWO ENDPOINTS OF THE SUB-             CADRE429
C  INTERVAL. IF TEST IS PASSED, ACCEPT *VINT* .                              CADRE430
   60 IF (LEVEL .GE. 4) WRITE (6,660) L                                     CADRE431
  660 FORMAT(I5,21X13HSTRAIGHT LINE)                                        CADRE432
      SLOPE = (FEND-FBEG)*2.                                                CADRE433
      FBEG2 = FBEG*2.                                                       CADRE434
      DO 61 I=1,4                                                           CADRE435
      DIFF = ABS(F(BEG+RN(I)*STEP) - FBEG2-RN(I)*SLOPE)                     CADRE436
      IF (DIFF .GT. TABTLM)            GO TO 72                             CADRE437
   61 CONTINUE                                                              CADRE438
      IF (LEVEL .GE. 3)  WRITE (6,667) BEG, END                             CADRE439
  667 FORMAT(27X43HINTEGRAND SEEMS TO BE STRAIGHT LINE BETWEEN              CADRE440
     *        E15.8,4H ANDE15.8)                                            CADRE441
                                       GO TO 80                             CADRE442
C-------------------------------------------------------------------------  CADRE443
C------  NOISE MAY BE DOMINANT FEATURE  -----------------------------------  CADRE444
C  ESTIMATE NOISE LEVEL BY COMPARING THE VALUE OF THE INTEGRAND AT          CADRE445
C  FOUR *RANDOMLY CHOSEN* POINTS WITH THE VALUE OF THE STRAIGHT LINE        CADRE446
C  INTERPOLATING THE INTEGRAND AT THE TWO ENDPOINTS. IF SMALL               CADRE447
C  ENOUGH, ACCEPT *VINT*.                                                   CADRE448
   70 IF (LEVEL .GE. 4) WRITE (6,670) L,T(1,LM1)                            CADRE449
  670 FORMAT(I5,E16.8,5X5HNOISE)                                            CADRE450
      SLOPE = (FEND-FBEG)*2.                                                CADRE451
      FBEG2 = FBEG*2.                                                       CADRE452
      I = 1                                                                 CADRE453
   71 DIFF = ABS(F(BEG+RN(I)*STEP) - FBEG2-RN(I)*SLOPE)                     CADRE454
   72 ERRER = AMAX1(ERRER,ASTEP*DIFF)                                       CADRE455
      IF (ERRER .GT. ERGOAL)           GO TO 91                             CADRE456
      I = I+1                                                               CADRE457
      IF (I .LE. 4)                    GO TO 71                             CADRE458
      IF (LEVEL .GE. 3) WRITE (6,671) BEG,END                               CADRE459
  671 FORMAT(15H NOISE BETWEEN ,E15.8,4H AND,E15.8)                         CADRE460
      IFLAG = 3                                                             CADRE461
C-------------------------------------------------------------------------  CADRE462
C--- INTEGRATION OVER CURRENT SUBINTERVAL SUCCESSFUL ----------------------  CADRE463
C  ADD *VINT* TO *CADRE* AND *ERRER* TO *ERROR*, THEN SET UP NEXT           CADRE464
C  SUBINTERVAL, IF ANY.                                                     CADRE465
   80 CADRE = CADRE + VINT                                                  CADRE466
      ERROR = ERROR + ERRER                                                 CADRE467
      IF (LEVEL .LT. 3)                GO TO 83                             CADRE468
      IF (LEVEL .LT. 5)                GO TO 82                             CADRE469
      DO 81 I=1,L                                                           CADRE470
   81 WRITE (6,692) I,(T(I,J),J=I,L)                                        CADRE471
   82 WRITE (6,682) VINT,ERRER,L,IT                                         CADRE472
  682 FORMAT(12H INTEGRAL IS,E16.8,7H, ERROR,E15.8,9H  FROM T(,             CADRE473
     *        I1,1H,,I1,1H))                                                CADRE474
C                                                                           CADRE475
   83 IF (RIGHT)                       GO TO 85                             CADRE476
      ISTAGE = ISTAGE - 1                                                   CADRE477
      IF (ISTAGE .EQ. 0)               RETURN                               CADRE478
C                                                                           CADRE479
      REGLAR = REGLSV(ISTAGE)                                               CADRE480
      BEG = BEGIN(ISTAGE)                                                   CADRE481
      END = FINIS(ISTAGE)                                                   CADRE482
      CUREST = CUREST - EST(ISTAGE+1) + VINT                                CADRE483
      IEND = IBEG - 1                                                       CADRE484
      FEND = TS(IEND)                                                       CADRE485
      IBEG = IBEGS(ISTAGE)                                                  CADRE486
                                       GO TO 94                             CADRE487
   85 CUREST = CUREST + VINT                                             CADRE488
      STAGE = STAGE*2.                                                   CADRE489
      IEND = IBEG                                                        CADRE490
      IBEG = IBEGS(ISTAGE)                                               CADRE491
      END = BEG                                                          CADRE492
      BEG = BEGIN(ISTAGE)                                                CADRE493
      FEND = FBEG                                                        CADRE494
      FBEG = TS(IBEG)                                                    CADRE495
                                            GO TO 5                      CADRE496
C----------------------------------------------------------------------  CADRE497
C--- INTEGRATION OVER CURRENT SUBINTERVAL IS UNSUCCESSFUL-------------   CADRE498
C  MARK SUBINTERVAL FOR FURTHER SUBDIVISION. SET UP NEXT SUBINTERVAL.    CADRE499
   90 REGLAR = .TRUE.                                                    CADRE500
   91 IF (ISTAGE .EQ. MXSTGE)               GO TO 950                    CADRE501
      IF (LEVEL  .LT. 5)                    GO TO 93                     CADRE502
      DO 92 I=1,L                                                        CADRE503
   92 WRITE (6,692) I,(T(I,J),J=1,L)                                     CADRE504
  692 FORMAT(I5,7E16.8/3E16.8)                                           CADRE505
   93 IF (RIGHT)                            GO TO 95                     CADRE506
      REGLSV(ISTAGE+1) = REGLAR                                          CADRE507
      BEGIN(ISTAGE) = BEG                                                CADRE508
      IBEGS(ISTAGE) = IBEG                                               CADRE509
      STAGE = STAGE/2.                                                   CADRE510
   94 RIGHT = .TRUE.                                                     CADRE511
      BEG = (BEG+END)/2.                                                 CADRE512
      IBEG = (IBEG+IEND)/2                                               CADRE513
      TS(IBEG) = TS(IBEG)/2.                                             CADRE514
      FBEG = TS(IBEG)                                                    CADRE515
                                            GO TO 6                      CADRE516
   95 NNLEFT = IBEG - IBEGS(ISTAGE)                                      CADRE517
      IF (IEND+NNLEFT .GE. MAXTS)           GO TO 900                    CADRE518
      III = IBEGS(ISTAGE)                                                CADRE519
      II = IEND                                                          CADRE520
      DO 96 I=III,IBEG                                                   CADRE521
      II = II + 1                                                        CADRE522
   96 TS(II) = TS(I)                                                     CADRE523
      DO 97 I=IBEG,II                                                    CADRE524
      TS(III) = TS(I)                                                    CADRE525
   97 III = III + 1                                                      CADRE526
      IEND = IEND + 1                                                    CADRE527
      IBEG = IEND - NNLEFT                                               CADRE528
      FEND = FBEG                                                        CADRE529
      FBEG = TS(IBEG)                                                    CADRE530
      FINIS(ISTAGE) = END                                                CADRE531
      END = BEG                                                          CADRE532
      BEG = BEGIN(ISTAGE)                                                CADRE533
      BEGIN(ISTAGE) = END                                                CADRE534
      REGLSV(ISTAGE) = REGLAR                                            CADRE535
      ISTAGE = ISTAGE + 1                                                CADRE536
      REGLAR = REGLSV(ISTAGE)                                            CADRE537
      EST(ISTAGE) = VINT                                                 CADRE538
      CUREST = CUREST + EST(ISTAGE)                                      CADRE539
                                            GO TO 5                      CADRE540
C----------------------------------------------------------------------  CADRE541
C--- FAILURE TO HANDLE GIVEN INTEGRATION PROBLEM----------------------   CADRE542
  900 IF (LEVEL .GE. 2) WRITE (6,6900) BEG, END                          CADRE543
 6900 FORMAT(37H TOO MANY FUNCTION EVALUATIONS AROUND/                   CADRE544
     *       10X,E15.8,4H AND,E15.8)                                     CADRE545
      IFLAG = 4                                                          CADRE546
                                            GO TO 999                    CADRE547
  950 IFLAG = 5                                                          CADRE548
```

```
      IF (LEVEL .LT. 2)                      GO TO 999                          CADRE549
      IF (LEVEL .LT. 5)                      GO TO 959                          CADRE550
      DO 958 I=1,L                                                              CADRE551
  958 WRITE (6,692) I,(T(I,J),J=1,L)                                            CADRE552
  959 WRITE (6,6959) BEG, END                                                   CADRE553
 6959 FORMAT(12X38HINTEGRAND SHOWS SINGULAR BEHAVIOUR OF                        CADRE554
     *       20HUNKNOWN TYPE BETWEENE15.8,4H ANDE15.8)                          CADRE555
  999 CADRE = CUREST + VINT                                                     CADRE556
                                             RETURN                             CADRE557
      END                                                                       CADRE558
```

```
         IDENT  SECOND
         ENTRY  SECOND
         SPACE  4
***      SECOND - RETURN ACCUMULATED CPU TIME IN SECONDS.
*        DAVID S. DODSON.  09/01/71.
         SPACE  4
***      FORTRAN CALL.
*
*
*        CALL SECOND (X)
*               X = VARIABLE FOR RETURN OF ACCUMULATED CPU TIME IN
*                             SECONDS AS A REAL NUMBER ACCURATE TO
*                             MILLISECONDS.  THIS ROUTINE USES THE
*                             *TIMEMS* REQUEST WHICH IS A NON-STANDARD
*                             FEATURE OF THE PURDUE OPERATING SYSTEM.
         TITLE  SECOND - RETURN ACCUMULATED CPU TIME IN SECONDS.
         TITLE  MAIN PROGRAM
SECOND   BSSZ   1             ENTRY/EXIT
         TIMEMS               READ ACCUMULATED CPU TIME IN MILLISECONDS
         SX1    1000          DIVIDE BY 1000
         PX1    X1
         NX1    X1
         PX6    X6
         NX6    X6
         FX6    X6/X1
         SA6    B1            STORE RESULT
         EQ     SECOND        RETURN
         SPACE  4
         END
```

```
      SUBROUTINE SPLINER(NOKNOT)
C                   NONLINEAR SPLINE APPROXIMATION
C     PROGRAM WRITTEN BY CARL DE BOOR AND JOHN RICE
C                             PURDUE UNIVERSITY
C     SUPPORTED BY THE NATIONAL SCIENCE FOUNDATION   GP-4052,GP-7163
C
C     PLEASE REPORT ANY CASES OF INOPERATION TO THE AUTHORS.
C                                   THANKS
C     ****              NUMERICAL ANALYSIS CONTROL                 ****
C            CONTROL PARAMETERS              FUNCTION
C              ITER                            NO. OF SWEEPS THRU OPT
C              BD (IN OPT)                     IMPROVEMENT NEEDED TO REPEAT
C              EPSERR( IN SWEEP)               IMPROVEMENT NEEDED TO REPEAT
C              DIST (IN OPT NEAR 30,80)        KEEPS KNOTS SEPARATED
C              INDLP                           NO. OF PASSES THRU OPT
C     THE FOLLOWING IS THE MAIN PROGRAM FOR VARYKNOT
C
      DIMENSION                     INFO(16)
C
C     COMMON INPUT SERVES AS INPUT TO FXDKNT
C                     SEE FXDKNT FOR DEFINITIONS OF VARIABLES
      COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE
C     COMMON OUTPUT SERVES AS OUTPUT FROM FXDKNT
C                     SEE FXDKNT FOR DEFINITIONS OF VARIABLES
      COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),
     *                 VORDL(28,2),KNCT,LMAX,INTERV
C
C     COMMON OTHER SERVES AS COMMUNICATION BETWEEN OPT,SWEEP AND HERE
C         LXI = NUMBER OF INTERIOR KNOTS,  LXI1 = LXI+1, LXI2 = LXI+2
C         G   = NUMERICAL CONTROL VARIABLE USED BETWEEN OPT AND SWEEP
C         CHANGE = DITTO
C         ERROR =  CURRENT VALUE OF THE L-2 ERROR - SQUARED
C         ACC   = DESIRED ACCURACY OF L-2 ERROR
C         XI(28)= ARRAY FOR KNOTS
      COMMON/ OTHER / LXI,LXI1,LXI2,G ,CHANGE,ERROR ,ACC,      XI(28)
C
C     ACC = .1 AND ITER = 4 TO 8 SEEM TO BE GOOD VALUES FOR TYPICAL USES
      ACC = .1
      ITER = 8
C
C  ***INFO IS SIMPLY AN IDENTIFICATION OF THE DATA***
C
C           READ IN NO. OF POINTS=LX AND THE DATA XX AND U
C  *** IF NOKNOT.GE.2, THEN READ IN LXI2=NOKNOT KNOTS***
C  ***      OTHERWISE PROGRAM CHOOSES LXI2 =-NOKNOT EQUISPACED KNOTS     ***
      LXI2 = IABS(NOKNOT)
C  IF *NOKNOT* IS .GT. 0, READ IN NOKNOT KNOTS (INCL.BOUNDARY POINTS.)
  601 FORMAT(6E12.6)
C
C        **CHECK ON GIVEN DATA
C                           AND FROM PRESENTING UNORDERED XX ARRAY
      IERROR = 0
      IF (LX .GE. LXI2+2 .AND. LX .LE. 100) GO TO 3
      WRITE(6,662) LX
      IERROR = 1
                                    GO TO 7
    3 IF (LXI2 .GE. 3 .AND. LXI2 .LE. 28)  GO TO 4
      WRITE(6,660) NOKNOT
      IERROR = 1
    4 DO 6 L=2,LX
         IF (XX(L)-XX(L-1))                  5,6,6
    5    WRITE(6,664) L,XX(L),U(L)
         IERROR = IERROR + 1
    6    CONTINUE
      IF (IERROR .LT. 1)                     GO TO 14
    7 WRITE(6,666) IERROR
      STOP
C
C       **INITIALIZE
   14 IF (NOKNOT .GT. 0)                     GO TO 30
C
C                WHEN NOKNOT IS NEG., INTRODUCE -NOKNOT EQUISPACED KNOTS
      XI(1) = XX(1)
      XI(LXI2) = XX(LX)
      DEL = (XX(LX) - XX(1))/FLOAT(LXI2-1)
      DO 26  J = 3,LXI2
   26 XI(J-1) = XI(J-2) + DEL
C
C                             SET UP INITIAL APPROXIMATION
   30 ADDXI(1) = XI(1)
      ADDXI(2) = XI(LXI2)
      LXI1 = LXI2-1
      LXI = LXI1-1
      MODE = 0
      JADD = LXI2
      DO 35 J = 3,LXI2
   35 ADDXI(J) = XI(J-1)
      ERROR = FXDKNT(0)
C      ***NOTE.   MODE HAS BEEN  SET EQUAL TO 1
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
      WRITE(6,611)  (L,XX(L),U(L),L=1,LX)
      WRITE(6,612) NOKNOT,ITER
      WRITE(6,900)  (XI(I), I=1,LXI2)
  900 FORMAT(28H KNOTS PRIOR TO OPTIMIZATION/(9F12.6))
C
C                                 OPTIMIZE KNOTS
      CALL SWEEP(ITER)
C
      WRITE(6,640)
  640 FORMAT(49X,22H***  FINAL OUTPUT  ***///)
      MODE = 1
      JADD = 0
      DUMB = FXDKNT(1)
      RETURN
C
  610 FORMAT(2I4,      /(7F12.8))
  611 FORMAT (11H GIVEN DATA//(I4,2F14.8))
  612 FORMAT(1H /32H NO. OF INITIAL          KNOTS =,I3/
     1        7H ITER =,I3)
  660 FORMAT(32H1KNOT CONTROL PARAMETER NOKNOT =I3,19H NOT WITHIN BOUNDS
     *.)
  662 FORMAT(26H NO. OF DATA POINTS, LX = I4,44H,NOT WITHIN THE BOUNDS A
     *BS(NOKNOT)+2 AND 100)
  664 FORMAT(12H DATA POINT I4,2F14.8,24H NOT IN ASCENDING ORDER.)
  666 FORMAT(23H *** CORRECT INDICATED I3,28H INPUT ERROR(S) AND RESTART
     *.)
```

```
      SUBROUTINE SWEEP(ITRR)
C
C        KVARY,I = INDEX OF KNOT       BEING VARIED
C        SUBROUTINE OPT(I) OPTIMIZES ITH INTERIOR KNOT
C
      COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE
      COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),
     *               VORDL(28,2),KNCT,LMAX,INTERV
      COMMON/ OTHER / LXI,LXI1,LXI2,C ,CHANGE,ERROR ,ACC,     XI(28)
C        AT ALL TIMES, ERROR CONTAINS (L2 ERROR)**2 OF CURRENT B.A.
C
      ITER = ITRR
C        **NEXT       CARDS SET NUMERICAL ANALYSIS CONTROLS
      EPSERR =    ACC/2.5
      CHANGE = .4*FLOAT(LXI)
C
   10 KVARY = LXI
      Q = CHANGE/FLOAT(LXI)
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT    ***
C***  WRITE (6,902) ITER,Q
C*902 FORMAT (8H ITER, Q  I5,E20.8)
      CHANGE = 0.
      PREVER = ERROR
      MODE = 2
      JADD = 0
      KNCT = KNOT - 1
      DUMB = FXDKNT(0)
   20 CONTINUE
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT    ***
C***  WRITE(6,900) KVARY
C*900 FORMAT(1H ///8H VARYING,I4,16HTH INTERIOR KNOT)
C***  WRITE(6,901) ERROR
C*901 FORMAT(16H SQ. OF L2-ERROR ,E16.6)
C
      CALL OPT(KVARY)
      KVARY = KVARY -1
      JADD = JADD + 1
      IF( JADD .LE. 1 )       GO TO 22
      K= JADD
      DO 21 I = 2,JADD
      K= K-1
   21 ADDXI(K+1) = ADDXI(K)
   22 ADDXI(1) = XI(KVARY + 2)
      KNCT = LXI1 - JADD
      MODE = 2
      DUMB = FXDKNT(0)
      IF( KVARY .NE. 0 )       GO TO 20
C        THE LAST CALL TO FXDKNT PRODUCES THE B.A. USING ALL KNOTS,
C        SINCE THEN ADDXI CONTAINS ALL KNOTS
      ERROR = DUMB
C     *** THE FOLLOWING TWO CARDS PRODUCE PRINTED OUTPUT OF L1,L2,L-INF
C**   JADD = 0
C**   DUMM = FXDKNT(2)
C
C       **IF CHANGE IN ERROR IS BIG ENOUGH MAKE ANOTHER SWEEP, ELSE QUIT
      IF (PREVER-ERROR .LE. EPSERR*PREVER)     GO TO 60
      ITER = ITER-1
C
```

```
C        **CHECK NUMBER OF PASSES THROUGH SWEEP
      IF(ITER.EQ.0) GO TO 40
      GO TO 10
   40 CONTINUE
C
C              IN FINAL VERSION GO TO 40, GO TO 60 ARE REPLACED BY RETURN
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
C***  WRITE(6,620)
      RETURN
   60 CONTINUE
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
C***  WRITE(6,610)
      RETURN
C*610 FORMAT(54H *** SWEEP DISCONTINUED - INSUFFICIENT CHANGE IN ERROR)
C*620 FORMAT (36H *** NO. OF ALLOWABLE SWEEPS USED UP)
      END
C
C***********************************************************************
C
```

```
      SUBROUTINE OPT(II)
C
C     I REFERS TO THE ITH INTERIOR KNOT
C     OPT FINDS THE OPTIMAL ITH KNOT BETWEEN THE I-1ST AND I+1ST KNOTS
C     THE REMAINING KNOTS ARE HELD FIXED.
C         INDLP = A BOUND ON THE NUMBER OF TRIES ALLOWED
C                 FOR IMPROVEMENT OF THE ITH KNOT
C         C = MULTIPLICATION FACTOR WHICH SHOULD DECREASE AS A
C             FUNCTION OF THE NO. OF SWEEPS THRU SWEEP
C             C IS ALTERED IN SWEEP
C
      COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE
      COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),
     *                VORDL(28,2),KNCT,LMAX,INTERV
      COMMON/ OTHER / LXI,LXI1,LXI2,C ,CHANGE,ERROR ,ACC,      XI(28)
C
      I = II
C       **NUMERICAL ANALYSIS PARAMETERS SET HERE
      INDLP=9
      BD = ACC*ERROR/FLOAT(LXI)
      DIST = .0625
      H = XI(I+2)-XI(I)
      ALOW = XI(I) + DIST*H
      AHIGH = XI(I+2) - DIST*H
      LPCNT= 0
      MODE = 3
C
C       **BEGIN SEARCH - FIND THREE VALUES FOR THE ITH KNOT
C         SUCH THAT L2-ERROR AT MIDDLE VALUE, A , IS LESS THAN
C         ERROR AT LEFT VALUE, ALEFT, AND AT RIGHT VALUE, ARIGHT
      A  = XI(I+1)
      E = FXDKNT(A)
      ALEFT = A + C*(XI(I)-A)
      ELEFT = FXDKNT(ALEFT)
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
C***  ARIGHT = 0.
C***  ERIGHT = 0.
C***  WRITE (6,900) ELEFT,E,ERIGHT,ALEFT,A,ARIGHT
      SGN = SIGN(1.,ELEFT-E)
      IF (SGN.GE.0.)                          GO TO 20
                                              GO TO 60
C
C       **SEARCHING FOR NEW KNOT TO THE RIGHT
   10 ALEFT = A
      ELEFT = E
      A = ARIGHT
      E = ERIGHT
   20 ARIGHT = A + C*(XI(I+2)-A)
C
C       **BUFFER TO PREVENT COALESCING OF KNOTS
   30 IF (AHIGH.GE.ARIGHT)                    GO TO 40
      AA = AHIGH
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
C**** WRITE(6,610) I
                                              GO TO 199
C
   40 ERIGHT = FXDKNT(ARIGHT)
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
```

```
C**** WRITE (6,900) ELEFT,E,ERIGHT,ALEFT,A,ARIGHT
      IF (E.LE.ERIGHT)                          GO TO 100
~~C~~
C       **CHECK TO STOP OPT
      IF(E -ERIGHT.LE.BD .OR. LPCNT .GT. INDLP )  GO TO 240
~~   50 LPCNT = LPCNT+1~~
      IF(SGN.GT.0.) GO TO 10
C
~~C        **SEARCHING FOR NEW KNOT TO THE LEFT~~
   60 ARIGHT = A
      ERIGHT = E
~~      A = ALEFT~~
      E = ELEFT
   70 ALEFT = A + Q*(XI(I)-A)
~~C~~
C
C       **BUFFER TO PREVENT COALESCING OF KNOTS
~~   80 IF (ALEFT.GE.ALOW)                        GO TO 90~~
      AA = ALOW
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
~~C**** WRITE(6,620) I~~
                                                GO TO 199
C
~~   90 ELEFT = FXOKNT(ALEFT)~~
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
C**** WRITE (6,900) ELEFT,E,ERIGHT,ALEFT,A,ARIGHT
~~      IF (E.LE.ELEFT)                           GO TO 100~~
C
C       **CHECK TO STOP OPT
~~      IF(E - ELEFT.LE.BD .OR. LPCNT .GT. INDLP )  GO TO 230~~
                                                GO TO 50
C
~~C        **REQUIRED 3 VALUES HAVE BEEN FOUND~~
C          FOLLOWING CODE FINDS PT. AT WHICH MIN OF PARABOLA CURVE PASSIN
C          THRU THE ERROR VALUES AT THE PTS ALEFT, A, ARIGHT  OCCURS
~~  100 DXLEFT = ALEFT - A~~
      DXRGHT = ARIGHT - A
      DYLEFT = (ELEFT-E)/DXLEFT
~~      DYRGHT = (ERIGHT-E)/DXRGHT~~
      DIFF = DYLEFT - DYRGHT
      IF (DIFF .EQ. 0.)                         GO TO 200
~~      DEL = .5/DIFF*(DXRGHT*DYLEFT-DXLEFT*DYRGHT)~~
      EPRED = E+DEL*(DYRGHT+(DXRGHT-DEL)/(ARIGHT-ALEFT)*DIFF)
      ABEST = A + DEL
~~      EBEST = FXOKNT(ABEST)~~
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
C***  WRITE (6,900) ELEFT,EBEST,ERIGHT,ALEFT,ABEST,ARIGHT
~~C~~
C       **DETERMINE WHETHER ABEST GIVES BEST APPRX AND MAKE APPROPRIATE
C          SWITCHING OF THE AI*S DEPENDING ON SIGN OF DEL
~~      IF (EBEST.LE.E)                           GO TO 130~~
      IF(DEL)110,200,120
  110 ALEFT = ABEST
~~      ELEFT = EBEST~~
      GO TO 170
  120 ARIGHT = ABEST
~~      ERIGHT = EBEST~~
      GO TO 170
  130 IF(DEL)140,200,150
  140 ARIGHT = A
      ERIGHT = E
      GO TO 160
  150 ALEFT = A
      ELEFT = E
  160 A = ABEST
      E = EBEST
C
C       **FOLLOWING       TESTS DETERMINE WHETHER OR NOT TO
C         REITERATE PARABOLA MINIMIZATION PHASE
  170 IF (ABS(EPRED-EBEST).LT.5.*BD)    GO TO 210
      IF(LPCNT.GT.INDLP)                GO TO 200
      LPCNT = LPCNT+1
                                        GO TO 100
C
  199 ETRY = FXOKNT(AA)
      IF (E.LT.ETRY)                    GO TO 200
      A = AA
      E = ETRY
  200 CHANGE = CHANGE + ABS(A -XI(I+1))/H
      XI(I+1) = A
      ERROR = E
C     *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT   ***
C***  WRITE (6,900) ELEFT,E,ERIGHT,ALEFT,A,ARIGHT
      RETURN
C
C           IN FINAL VERSION GO TO 210, IS REPLACED BY GO TO 200
  210 CONTINUE
C     *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT   ***
C***  WRITE(6,640) LPCNT
      GO TO 200
  230 A = ALEFT
      E = ELEFT
C     *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT   ***
C***  WRITE(6,640) LPCNT
      GO TO 200
  240 A = ARIGHT
      E = ERIGHT
C     *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT   ***
C***  WRITE(6,640) LPCNT
      GO TO 200
C*610 FORMAT(46H *** OPT DISCONTINUED - KNOT BEING OPTIMIZED (,I2,35H) M
C****1OVED TOO CLOSE TO RIGHT NEIGHBOR)
C*620 FORMAT(46H *** OPT DISCONTINUED - KNOT BEING OPTIMIZED (,I2,34H) M
C****1OVED TOO CLOSE TO LEFT NEIGHBOR)
C*640 FORMAT(24H *** OPT DISCONTINUED AT,I4,31H - INSUFFICIENT CHANGE IN
C*** * ERROR)
C*900 FORMAT(25H PARABOLA - ERROR VALUES ,3E20.6/12X,13HAI VALUES    ,
C*** 1        3E20.6)
      END

C----------------------------------------------------------------------
```

```
      FUNCTION FXOKNT (ARG)                                             SPLIN377
C                          THE FUNCTION RETURNS THE SQUARE OF THE L2-ERROR SPLIN378
      ~~LOGICAL MODE3                                                     SPLIN379~~
C**   IT MAY BE NECESSARY ON SOME SYSTEMS TO MENTION ALL COMMON BLOCKS  SPLIN380
C     LISTED HERE IN THE PROGRAM CALLING *FXOKNT*, TOO, TO INSURE THAT  SPLIN381
~~C     THE INFO IN THESE BLOCKS DOES NOT DIE BETWEEN CALLS TO *FXOKNT*   SPLIN382~~
      COMMON / WANDT / TREND(100),TRPZWT(100),PRINT(100)                SPLIN383
      COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE                SPLIN384
~~C           U(L) = FCT TO BE APPR AT  XX(L), L=1,LX.                    SPLIN385~~
C                XX(L) IS ASSUMED TO BE NONDECREASING WITH L            SPLIN386
C           ADDXI(I) = I-TH KNOT TO BE INTRODUCED, I=1,JADD             SPLIN387
~~C           MODE = 0,1,2,3 . SEE COMMENTS BELOW ( AND IN NUBAS)         SPLIN388~~
      COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),        SPLIN389
     *                VORDL(28,2),KNOT,LMAX,INTERV                      SPLIN390
~~C           UERROR(L) = ERROR OF B.L2 A. TO U, L=1,LX                   SPLIN391~~
C           KNOT = CURRENT NO. OF KNOTS (INCL BDRY KNOTS)               SPLIN392
C           INTERV = KNOT - 1  = CURRENT NO. OF INTERVALS (POL.PIECES)  SPLIN393
~~C           XIL(K),K=1,KNOT, CURRENT (ORDERED) SET OF KNOTS             SPLIN394~~
C           THE MAXIMUM ERROR OCCURS AT XX(LMAX)                        SPLIN395
C        IF ARG=1, FCTL(L) CONTAINS THE CURRENT B.AB TO U AT XX(L)      SPLIN396
~~C           COEFL(I,.) CONTAINSTHE POL.COEF. ON I-TH INTERVAL FOR B.A.  SPLIN397~~
C           VORDL(I,.) CONTAINS VALUE AND DERIV. OF B.A. AT XIL(I)      SPLIN398
      COMMON/ BASIS /FCT(100,30),VORD(30,28,2),BC(30),ILAST             SPLIN399
~~C**   A CHANGE IN THE COLUMN LENGTH OF *FCT* FORCES CHANGE IN ST.NO.69  SPLIN400~~
C     IN *NUBAS* .                                                      SPLIN401
C           FCT (L,M) = BASIS FCT M AT XX(L)                            SPLIN402
~~C         VORD(M,K,L) CONTAINS THE ORDS (L=1) AND SLOPES (L=2) OF FCT M SPLIN403~~
C           AT THE KNOT INTRODUCED AS K-TH. CORRELATION TO ORDERING OF  SPLIN404
C           KNOTS BY SIZE IS DONE VIA IORDER, I.E., ORD AND SLOPE AT    SPLIN405
~~C           XIL(K) ARE IN VORD(M,IORDER(K),.).                          SPLIN406~~
C           BC(I) = COORDINATE OF U (AND OF B.A. TO U) WRTO I-TH O.N.FCTSPLIN407
C           ILAST = CURRENT NO. OF BASIS FCTNS                          SPLIN408
      ~~COMMON/ LASTB /IORDER(28),INSIRT(30),XKNOT                        SPLIN409~~
C           THE FCT ILAST (TO BE) INTRODUCED LAST HAS ADDITIONAL KNOT   SPLIN410
C           XKNOT, THE KNOT JUST INTRO-                                 SPLIN411
~~C           DUCED HAS INDEX INSERT IN XIL,INSERT IS SAVED IN INSIRT(ILASSPLIN412~~
C           FOR POSSIBLE REPLACEMENT OF KNOTS LATER ON (SEE MODE=2,3).  SPLIN413
C      ***LOCAL VARIABLES                                               SPLIN414
      ~~COMMON /LOCAL/ XSCALE,KNOTSV,ERBUT1,CUBERR(100),WEIGHT(100),MODE3 SPLIN415~~
C           XSCALE = XX(LX) - XX(1), USED TO NORMALIZE INNER PRODUCT    SPLIN416
C                  = LENGTH OF THE INTERVAL OF INTEGRATION              SPLIN417
~~C           KNOTSV = NO. OF KNOTS USED IN MOST RECENT CALL TO FXOKNT    SPLIN418~~
C           ERBUT1 = SQ. OF L2-ERROR OF APPR USING ALL BUT THE ONE      SPLIN419
C                    KNOT BEING VARIED ( USED IN MODE = 3)              SPLIN420
~~C           CUBERR = UERROR OF B.A. BY CUBIC POL-S (NEEDED FOR MODE = 2)SPLIN421~~
C           MODE3 = TRUE OR FALSE DEP. ON WHETHER PREV. CALL WAS IN     SPLIN422
C                   MODE=3 OR NOT                                       SPLIN423
      ~~EQUIVALENCE (IPRINT,CHANGE)                                       SPLIN424~~
C           ARG IS EITHER FIXED POINT (MODE.NE.3) TO PICK PRINT-OUT OPTISPLIN425
C           OR IS FLOATING POINT (MODE=3) TO GIVE NEW VALUE OF KNOT VARISPLIN426
      ~~CHANGE = ARG                                                      SPLIN427~~
      IF (MODE.GT.0)                 GO TO 29                           SPLIN428
C-----------                                                            SPLIN429
~~C        *** MODE=0* COMPUTE BASIS FCT 1 THROUGH 4 AND B.A. TO U WRTO THESPLIN430~~
C        THEN SET MODE = 1 AND PUT UERROR INTO U.                       SPLIN431
      XSCALE = XX(LX) - XX(1)                                           SPLIN432
      ~~DO 10 I=5,30                                                      SPLIN433~~
   10 INSIRT(I) = 0                                                     SPLIN434
```

```
      DO 11 L=1,LX
      UERROR(L) = U(L)
      ~~TREND(L) = T(XX(L))~~
   11 WEIGHT(L) = W(XX(L))
      DO 12 L=2,LX
   ~~12 TRPZWT(L-1) = (XX(L)-XX(L-2))/2.*WEIGHT(L-1)~~
      TRPZWT(1) = (XX(2)-XX(1))/2.*WEIGHT(1)
      TRPZWT(LX) = (XX(LX)-XX(LX-1))/2.*WEIGHT(LX)
~~C~~
      XIL(1) = ADDXI(1)
      XIL(2) = ADDXI(2)
      ~~IORDER(1) = 1~~
      IORDER(2) = 2
      KNOT = 2
      ~~INTERV = 1~~
      DO 19 I=1,4
      ILAST = I
      ~~CALL NUBAS~~
      DO 19 L=1,LX
   19 UERROR(L) = UERROR(L) - BC(I)*FCT(L.I)
~~C~~
      MODE = 1
      DO 20 L = 1,LX
   ~~20 CUBERR(L) = UERROR(L)~~
C        IF (JADD.LE.2), ONLY B.A. BY CUBICS IS COMPUTED
C         OTHERWISE, ADDXI(I), I.GT.2, CONTAINS ADDITIONAL KNOTS
      ~~JADD = JADD - 2~~
      IF (JADD.LE.0)                   GO TO 60
      DO 21 I=1,JADD
   ~~21 ADDXI(I) = ADDXI(I+2)~~
                                       GO TO 51
C-----------
   ~~29                                    GO TO (40,40,30),MODE~~
C-----------
C        *** MODE=3 *** MERELY REPLACE THE LAST KNOT INTRODUCED BY
~~C                       CHANGE AND RECOMPUTE L2 ERROR.  CHANGE ENTERS~~
C                       VIA THE ARGUMENT JPRINT = CHANGE.
C                       THIS MODE SHOULD BE USED FOR
~~C                               MINIMIZING THE L2-ERROR  WRTO THE KNOT~~
C                       INTRODUCED LAST AS IT MINIMIZES THE COMP WORK
C                  IF MODE3 = TRUE (I.E., THE PRECEDING CALL TO FXDKNT
~~C                       WAS IN MODE=3),THE PROGR WILL ASSUME THAT CHANGE~~
C                       HAS THE SAME ORDER REL TO THE OTHER KNOTS AS THE
C                       PREV INTRODUCED VALUE FOR KNOT. OTHERWISE
~~C                  IF MODE3=FALSE(THE PRECEDING CALL WAS IN SOME OTHER M~~
C                       , A FCT IS ADDED WITH CHANGE AS THE ADD. KNOT.
C                       UERROR IS ASSUMED TO CONTAIN ERROR OF B.A. TO U
~~C                       ALL PREV FCTNS. **NOTE** IF THE NEXT CALL TO FXD~~
C                       IS IN A MODE OTHER THAN 3, THE CHANGE PROPOSED
C                       NOW WILL BE MADE PERMANENT.
   ~~30 XKNOT = CHANGE~~
      IF (MODE3)                       GO TO 35
      MODE3 = .TRUE.
      ~~ERBUT1 = FXDKNT~~
      MODE = 2
      CALL NUBAS
      ~~KNOTSV = KNOT~~
      MODE = 3
```

```
                                    GO TO 36                          SPLIN493
   35 CALL NUBAS                                                      SPLIN494
   36 FXDKNT = ABS(ERBUT1 - BC(ILAST)/XSCALE*BC(ILAST))               SPLIN495
                                    RETURN                            SPLIN496
C----------                                                           SPLIN497
C        ***MODE=1,2***   RETAIN THE FIRST KNOT KNOTS INTRODUCED EARLIER  SPLIN498
C                         (HENCE THEIR CORRESP FCTNS) BUT REPLACE FURTHER SPLIN499
C                         FCTNS (IF ANY) BY FCTNS HAVING ADDITIONAL       SPLIN500
C                         KNOTS ADDXI(I),I=1,JADD) HENCE                  SPLIN501
C                         IF KNOT.LT.KNOTSV(=NO.OF KNOTS USED IN PREV CALLSPLIN502
C        40 THROUGH 49 RESTORES ARRAYS IORDER,XIL, UERROR TO THE STATE OSPLIN503
C        ILAST = KNOT + 2 ,  INVERTING THE ACTION OF DO 11 ... TO 14 IN SPLIN504
   40 IF (KNOT.LT.KNOTSV)           GO TO 42                          SPLIN505
      KNOT = KNOTSV                                                   SPLIN506
      IF (.NOT.MODE3)               GO TO 50                          SPLIN507
      DO 41 L=1,LX                                                    SPLIN508
   41 UERROR(L) = UERROR(L) - BC(ILAST)*FCT(L,ILAST)                  SPLIN509
                                    GO TO 49                          SPLIN510
   42 DO 43 L=1,LX                                                    SPLIN511
   43 UERROR(L) = CUBERR(L)                                           SPLIN512
      IF (KNOT.LE.2)                GO TO 48                          SPLIN513
      IDUM = KNOT + 1                                                 SPLIN514
      DO 45 IO=IDUM,KNOTSV                                            SPLIN515
      INSERT = INSIRT(ILAST)                                          SPLIN516
      ILM3 = ILAST - 3                                                SPLIN517
      DO 44 K=INSERT,ILM3                                             SPLIN518
      IORDER(K) = IORDER(K+1)                                         SPLIN519
   44 XIL(K) = XIL(K+1)                                               SPLIN520
   45 ILAST = ILAST-1                                                 SPLIN521
      DO 47 I=5,ILAST                                                 SPLIN522
      DO 47 L=1,LX                                                    SPLIN523
   47 UERROR(L) = UERROR(L) - BC(I)*FCT(L,I)                          SPLIN524
                                    GO TO 49                          SPLIN525
   48 XIL(2) = XIL(ILAST-2)                                           SPLIN526
      IORDER(2) = 2                                                   SPLIN527
      KNOT = 2                                                        SPLIN528
   49 IF (JADD.GT.0)                 GO TO 51                          SPLIN529
      ILAST = KNOT + 2                                                SPLIN530
      INTERV = KNOT - 1                                               SPLIN531
                                    GO TO 60                          SPLIN532
C                                                                     SPLIN533
C        ***MODE=1,2***   ADD JADD BASIS FCTNS, I.E., FOR IO=1,JADD,      SPLIN534
C                         CONSTRUCT FCT ILAST WITH ONE MORE KNOT, VIZ.    SPLIN535
C                         XKNOT=ADDXI(IO), THAN THE PREVIOUS LAST FCT,    SPLIN536
C                         ORTHONORMALIZE IT OVER ALL PREVIOUS  FCTNS, THENSPLIN537
C                         COMPUTE THE COORDINATE BC(ILAST) OF U WRTO IT,  SPLIN538
C                         SUBTRACT OUT ITS COMPONENT FROM UERROR.         SPLIN539
   50 IF (JADD.LE.0)                GO TO 61                          SPLIN540
   51 DO 52 IO=1,JADD                                                 SPLIN541
      XKNOT = ADDXI(IO)                                               SPLIN542
      CALL NUBAS                                                      SPLIN543
      DO 52 L=1,LX                                                    SPLIN544
   52 UERROR(L) = UERROR(L) - BC(ILAST)*FCT(L,ILAST)                  SPLIN545
C                                                                     SPLIN546
   60 FXDKNT= DOT(31,2)/XSCALE                                        SPLIN547
      KNOTSV = KNOT                                                   SPLIN548
   61 MODE3 = .FALSE.                                                 SPLIN549
      IF (IPRINT.EQ.0)               RETURN                           SPLIN550
```

```
C          VARIOUS PRINTING IS DONE DEP ON THE ARG = IPRINT                SPLIN551
                                       GO TO (70,80,90),IPRINT             SPLIN552
~~C                                                                          SPLIN553~~
C         COMPUTE COEFFICIENTS OF B.A. AND PRINT                           SPLIN554
C      ****              BEST APPROXIMATION PRINTOUT               ****    SPLIN555
~~C         FORMAT IS                                                        SPLIN556~~
C               KNOTS XI(J)           CUBIC COEFFICIENTS P(I,J) IN         SPLIN557
C                                     INTERVAL (XI(J), XI(J+1))            SPLIN558
~~C                            ERROR CURVE (SCALED)                          SPLIN559~~
C                                                                          SPLIN560
C        THE FOLLOWING FORTRAN CODE FINDS VALUES AT X OF THE               SPLIN561
~~C        APPROXIMATION FROM THIS OUTPUT----                                SPLIN562~~
C                     I=LXI2                                               SPLIN563
C                   1 A=X-XI(I)                                            SPLIN564
~~C                     IF(A) 2,4,4                                          SPLIN565~~
C                   2 I=I-1                                                SPLIN566
C                     IF(I) 3,3,1                                          SPLIN567
~~C                   3 I=1                                                  SPLIN568~~
C                   4 V=P(I,1)+A*(P(I,2)+A*(P(I,3)+A*P(I,4)))              SPLIN569
C ..... FOR A SUBPROGRAM USE COMMON/OUTPUT/..., COMMON/OTHER/...,
~~C ..... THE P BECOME COEFL(I,1)COEFL(I,2),COEFL(I,3),COEFL(I,4)~~
C                                                                          SPLIN570
   70 WRITE(6,610)                                                         SPLIN571
~~      DO 72  I=1,KNOT                                                      SPLIN572~~
      ILOC = IORDER(I)                                                     SPLIN573
      DO 72  L=1,2                                                         SPLIN574
~~   72 VORDL(I,L) = -ARITH1(0.,ILAST,BC,1,VORD(1,ILOC,L),1)                 SPLIN575~~
C  SEE COMMENT IN *DOT* ABOUT THE *ARITH1* ROUTINE.                        SPLIN576
      CALL EVAL                                                            SPLIN577
~~      DO 73 I=1,INTERV                                                     SPLIN578~~
      WRITE(6,620) I,XIL(I)                                                SPLIN579
   73 WRITE (6,630) (J,COFFL(I  ,J),J=1,4)                                 SPLIN580
~~      WRITE (6,620) KNOT,XIL(KNOT)                                         SPLIN581~~
  610 FORMAT(42X,5HKNOTS,22X,18HCUBIC COEFFICIENTS//)                      SPLIN582
  620 FORMAT(35X, 3HXI(, I2, 3H) =, F12.6)                                 SPLIN583
~~  630 FORMAT(67X,2HC(,I1,3H) =,E16.6)                                      SPLIN584~~
C                                                                          SPLIN585
C         **COMPUTE L2, L1, MAX ERRORS AND PRINT                           SPLIN586
~~   80 ERRL2 = SQRT(FXDKNT)                                                 SPLIN587~~
      ERRL1 = 0.                                                           SPLIN588
      ERRL99= 0.                                                           SPLIN589
~~      DO 82 L=1,LX                                                         SPLIN590~~
       DIF = ABS(UERROR(L)*WEIGHT(L))                                      SPLIN591
      IF(ERRL99.GT.DIF)                     GO TO 81                       SPLIN592
~~      LMAX = L                                                             SPLIN593~~
      ERRL99 = DIF                                                         SPLIN594
   81 ERRL1 = ERRL1+ DIF                                                   SPLIN595
~~   82 CONTINUE                                                             SPLIN596~~
      ERRL1 = ERRL1/FLOAT(LX)                                              SPLIN597
      WRITE(6,623) ERRL2, ERRL1, ERRL99,XX(LMAX)                           SPLIN598
~~C     *** THE FOLLOWING CARD IS TEMPORARY                                  SPLIN599~~
      GO TO (90,96,96),IPRINT                                              SPLIN600
C                                                                          SPLIN601
~~C        **  SCALE ERROR CURVE AND PRINT                                   SPLIN602~~
   90 IE = 0                                                               SPLIN603
      SCALE = 1.                                                           SPLIN604
~~      IF (ERRL99.GE.10.)                    GO TO 92                       SPLIN605~~
      DO 91 IE=1,9                                                         SPLIN606
      SCALE = SCALE*10.                                                 SPLIN607
      IF (ERRL99*SCALE.GE.10.)           GO TO 92                       SPLIN608
   91 CONTINUE                                                          SPLIN609
   92 DO 93 L=1,LX                                                      SPLIN610
   93 PRINT (L) = UERROR(L)*SCALE                                       SPLIN611
                                         GO TO (94,95,95),IPRINT        SPLIN612
   94 WRITE (6,621) IE,(L,XX(L),FCTL(L),PRINT(L),L=1,LX)                SPLIN613
                                         GO TO 96                       SPLIN614
   95 WRITE (6,622) IE,(L,XX(L),PRINT(L),L=1,LX)                        SPLIN615
   96                                    RETURN                         SPLIN616
  621 FORMAT(1H //45X,36HAPPROXIMATION AND SCALED ERROR CURVE/38X,      SPLIN617
     *10HDATA POINT,7X,13HAPPROXIMATION,3X,16HDEVIATION X 10E+,I1/      SPLIN618
     *(31X,I4,F16.8,F16.8,F17.6))                                       SPLIN619
  622 FORMAT(1H //53X, 11HERROR CURVE/38X, 10HDATA POINT, 23X,          SPLIN620
     116HDEVIATION X 10E+,I1/(31X,I4,F16.8,16X,F17.6))                  SPLIN621
  623 FORMAT(1H ///40X20HLEAST SQUARE ERROR =,E20.6/                    SPLIN622
     1            40X20HAVERAGE ERROR      =,E20.6/                     SPLIN623
     2            40X20HMAXIMUM ERROR      =,E20.6,3H AT,F12.6///)      SPLIN624
      END                                                               SPLIN625
C                                                                       SPLIN626
C***********************************************************************SPLIN627
C                                                                       SPLIN628
```

```
      SUBROUTINE INTERP                                                 SPLIN629
C                                                                       SPLIN630
C           COMPUTE THE SLOPES VORDL(I,2), I=2,KNOT-1 AT INTERIOR       SPLIN631
C           KNOTS OF CUBIC SPLINE FOR GIVEN VALUES VORDL(I,1),I=1,KNOT, SPLIN632
C           AT ALL  THE KNOTS AND GIVEN BOUNDARY DERIVATIVES            SPLIN633
      DIMENSION D(28), DIAG(28)                                         SPLIN634
      COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),        SPLIN635
     *                   VORDL(28,2),KNOT,LMAX,INTERV                   SPLIN636
      DATA DIAG(1),D(1)/1.,0./                                          SPLIN637
      DO 10 M=2,KNOT                                                    SPLIN638
      D(M) = XIL(M) - XIL(M-1)                                          SPLIN639
   10 DIAG(M) = (VORDL(M,1)-VORDL(M-1,1))/D(M)                          SPLIN640
      DO 20 M=2,INTERV                                                  SPLIN641
      VORDL(M,2) = 3.*(D(M)*DIAG(M+1) + D(M+1)*DIAG(M))                 SPLIN642
   20 DIAG(M) = 2.*(D(M)+D(M+1))                                        SPLIN643
      DO 30 M=2,INTERV                                                  SPLIN644
      G = -D(M+1)/DIAG(M-1)                                             SPLIN645
      DIAG(M) = DIAG(M) + G*D(M-1)                                      SPLIN646
   30 VORDL(M,2) = VORDL(M,2) + G*VORDL(M-1,2)                          SPLIN647
      NJ = KNOT                                                         SPLIN648
      DO 40 M=2,INTERV                                                  SPLIN649
      NJ = NJ - 1                                                       SPLIN650
   40 VORDL(NJ,2) = (VORDL(NJ,2) - D(NJ)*VORDL(NJ+1,2))/DIAG(NJ)        SPLIN651
                                        RETURN                          SPLIN652
      END                                                               SPLIN653
C                                                                       SPLIN654
C***********************************************************************SPLIN655
C                                                                       SPLIN656
```

```
      FUNCTION DOT (M,INDEX)                                            SPLIN657
C         COMPUTE INNER PRODUCT OF FCT M WITH FCT ILAST (INDEX=1) OR    SPLIN658
C     ~~UERROR (INDEX=2)~~                                              ~~SPLIN659~~
      COMMON / WANDT / TREND(100),TRPZWT(100),G(100)                    SPLIN660
      COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE                SPLIN661
      ~~COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),~~    ~~SPLIN662~~
     *                    VORDL(28,2),KNOT,LMAX,INTERV                  SPLIN663
      COMMON/ BASIS /FCT(100,30),VORD(30,28,2),BC(30),ILAST             SPLIN664
                                          ~~GO TO (10,30),INDEX~~       ~~SPLIN665~~
   10 IF (M.EQ.ILAST)                     GO TO 20                      SPLIN666
      DO 11 L=1,LX                                                      SPLIN667
   ~~11 G(L) = FCT(L,M)*FCTL(L)~~                                       ~~SPLIN668~~
                                          GO TO 80                      SPLIN669
   20 DO 21 L=1,LX                                                      SPLIN670
   ~~21 G(L) = FCTL(L)*FCTL(L)~~                                        ~~SPLIN671~~
                                          GO TO 80                      SPLIN672
   30 IF (M.EQ.31)                        GO TO 40                      SPLIN673
      ~~DO 31 L=1,LX~~                                                  ~~SPLIN674~~
   31 G(L) = FCTL(L)*UERROR(L)                                          SPLIN675
                                          GO TO 80                      SPLIN676
   ~~40 DO 41 L=1,LX~~                                                  ~~SPLIN677~~
   41 G(L) = UERROR(L)*UERROR(L)                                        SPLIN678
C                                                                       SPLIN679
~~C  EFFICIENTLY PROGRAMMED DOUBE PRECISION ACCUMULATION OF SCALAR~~   ~~SPLIN680~~
C  PRODUCTS IS CALLED FOR HERE.  AT PURDUE, WE USE                      SPLIN681
C         D  =  ARITH1(C,N,A,IA,B,IB)                                   SPLIN682
~~C  WHICH RETURNS THE VALUE OF~~                                       ~~SPLIN683~~
C         D  =  C - SUM(A(1+J*IA) * B(1+J*IB), J=0,...,N-1)             SPLIN684
C                                                                       SPLIN685
   ~~80 DOT = -ARITH1(0.,LX,G,1,TRPZWT,1)~~                             ~~SPLIN686~~
                                          RETURN                        SPLIN687
      END                                                               SPLIN688
~~C~~                                                                   ~~SPLIN689~~
C***********************************************************************SPLIN690
C                                                                       SPLIN691
```

```
      SUBROUTINE EVAL
C        COMPUTE POL. COEFF COEFL(I,K) OF FCT ILAST FROM VORDL,
C        THEN COMPUTE FCTL(L) = (FCT ILAST)*TREND AT XX(L),L=1,LX
C
      COMMON / WANDT / TREND(100),TRPZWT(100),G(100)
      COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE
      COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),
     *                   VORDL(28,2),KNOT,LMAX,INTERV
      DO 10 I=1,INTERV
      COEFL(I,1) = VORDL(I,1)
      COEFL(I,2) = VORDL(I,2)
      DX = XIL(I+1) - XIL(I)
      DUM1 = (VORDL(I+1,1)-VORDL(I,1))/DX
      DUM2 =   VORDL(I,2)+VORDL(I+1,2)-2.*DUM1
      COEFL(I,3) = (DUM1-DUM2-VORDL(I,2))/DX
   10 COEFL(I,4) = DUM2/DX/DX
C
      J = 1
      ISWTCH = 1
      DO 20 L=1,LX
                                      GO TO (11,13),ISWTCH
   11 IF (J.EQ.INTERV)                GO TO 12
      IF (XX(L).LT.XIL(J+1))          GO TO 13
      J = J + 1
                                      GO TO 11
   12 ISWTCH = 2
   13 DX = XX(L) - XIL(J)
   20 FCTL(L) = (COEFL(J,1)+DX*(COEFL(J,2)+DX*(COEFL(J,3)
     *                                  +DX*COEFL(J,4))))*TREND(L)
                                      RETURN
      END
C
C*********************************************************************
C
```

```
      SUBROUTINE NUBAS                                                  SPLIN726
      COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE                SPLIN727
      ~~COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),~~      ~~SPLIN728~~
     *                      VORDL(28,2),KNOT,LMAX,INTERV                SPLIN729
      COMMON/ BASIS /FCT(100,30),VORD(30,28,2),BC(30),ILAST             SPLIN730
      ~~COMMON/ LASTB /IORDER(28),INSIRT(30),XKNOT~~                      ~~SPLIN731~~
C             COEF(IC,.) CONTAINS THE POL COEFFICIENTS OF FCT M FOR INTER-SPLIN732
C             VAL TO THE RIGHT OF XI(IC), IC=ICM,ICM+M-3,               SPLIN733
~~C             WITH ICM = M*(M-7)/2 + 10 (WITH OBVIOUS MODS FOR M.LE.4)    SPLIN734~~
C             THE FCT ILAST (TO BE) INTRODUCED LAST, HAS ITS VALUES AT THESPLIN735
C             THE POINTS XX(L) IN FCTL(L),          HAS FIRST INDEX ICLASSPLIN736
~~C             IN COEF AND XI, HAS ADDITIONAL KNOT XKNOT, THE KNOT KNOTS   SPLIN737~~
C             FOR IT ARE CONTAINED, IN INCREASING ORDER, IN XIL,ITS COR-  SPLIN738
C             RESPONDING ORDS AND SLOPES ARE IN VORDL, THE KNOT JUST INTROSPLIN739
~~C             DUCED HAS INDEX INSERT IN XIL,INSERT IS SAVED IN INSIRT(ILASSPLIN740~~
C             FOR POSSIBLE REPLACEMENT OF KNOTS LATER ON (SEE MODE=2,3).  SPLIN741
      DIMENSION TEMP(30),XI(361),COEF(361,4)                            SPLIN742
      ~~IF (MODE.GT.0)                     GO TO 8~~                        ~~SPLIN743~~
C--------***CONSTRUCT FCT ILAST FOR ILAST.LE.4                          SPLIN744
      XI(ILAST) = XIL(1)                                                SPLIN745
      ~~ICLAST = ILAST~~                                                    ~~SPLIN746~~
      ILM1 = ILAST-1                                                    SPLIN747
      IF(ILAST.GT.2)                      GO TO 7                       SPLIN748
      ~~IF(ILAST.EQ.2)                      GO TO 6~~                       ~~SPLIN749~~
C           FIRST BASIS FCT IS A CONSTANT                               SPLIN750
      VORDL(1,1)=1.                                                     SPLIN751
      ~~VORDL(2,1)=1.~~                                                     ~~SPLIN752~~
      VORDL(1,2)=0.                                                     SPLIN753
      VORDL(2,2)=0.                                                     SPLIN754
                                          ~~GO TO 67~~                      ~~SPLIN755~~
C             SECOND BASIS FCT IS A STRAIGHT LINE                       SPLIN756
    6 VORDL(2,2) = VORDL(1,1)/(XIL(2) - XIL(1))*2.                      SPLIN757
      ~~VORDL(1,2) =-VORDL(2,2)~~                                           ~~SPLIN758~~
C                                                                       SPLIN759
    7 VORDL(2,1) = - VORDL(2,1)                                         SPLIN760
      ~~VORDL(2,2) = - VORDL(2,2)~~                                         ~~SPLIN761~~
                                          GO TO 59                      SPLIN762
C--------                                                               SPLIN763
    ~~8~~                                     ~~GO TO (10,10,14),MODE~~         ~~SPLIN764~~
C--------***SET UP CONSTANTS DEP.ON ILAST. INSERT NEW KNOT  INTO XIL     SPLIN765
C             AND UPDATE VORD FOR FCT M,M=1,ILAST-1                     SPLIN766
   ~~10 KNOT = KNOT + 1~~                                                   ~~SPLIN767~~
      ILAST = KNOT + 2                                                  SPLIN768
      ICLAST = ILAST*(ILAST-7)/2 + 10                                   SPLIN769
      ~~ILM1 = ILAST-1~~                                                    ~~SPLIN770~~
      INTERV = KNOT - 1                                                 SPLIN771
      DO 11 INSERT=2,INTERV                                             SPLIN772
      ~~IF (XKNOT.LT.XIL(INSERT))           GO TO 12~~                      ~~SPLIN773~~
   11 CONTINUE                                                          SPLIN774
                                          GO TO 95                      SPLIN775
   ~~12 IF (XKNOT.LE.XIL(INSERT-1))         GO TO 95~~                      ~~SPLIN776~~
      IO = KNOT                                                         SPLIN777
      DO 13 L=INSERT,INTERV                                             SPLIN778
      ~~IO = IO - 1~~                                                       ~~SPLIN779~~
      XIL(IO+1) = XIL(IO)                                               SPLIN780
   13 IORDER(IO+1) = IORDER(IO)                                         SPLIN781
      ~~IORDER(INSERT) = KNOT~~                                             ~~SPLIN782~~
C                                                                       SPLIN783
```

```
   14 XIL(INSERT) = XKNOT
      DX = XKNOT - XIL(1)
      ~~DO 15 I=1,4~~
      VORD(I,KNOT,1)=COEF(I,1)+DX*(COEF(I,2)+DX*(COEF(I,3)
     *                                  +DX*COEF(I,4)))
   ~~15 VORD(I,KNOT,2)=COEF(I,2)+DX*(2.*COEF(I,3)+DX*3.*COEF(I,4))~~
      IF(ILM1.LT.5) GO TO 20
      ID = 4
      ~~IBOUND = 4~~
      DO 19 I=5,ILM1
      ID = ID + I - 4
      ~~IBOUND = IBOUND + I - 3~~
   17 IF (ID.EQ.IBOUND)                GO TO 18
      IF (XKNOT.LT.XI(ID+1))           GO TO 18
      ~~ID = ID + 1~~
                                       GO TO 17
   18 DX = XKNOT - XI(ID)
      ~~VORD(I,KNOT,1)=COEF(ID,1)+DX*(COEF(ID,2)+DX*(COEF(ID,3)~~
     *                                      +DX*COEF(ID,4)))
   19 VORD(I,KNOT,2)=COEF(ID,2)+DX*(COEF(ID,3)*2.+DX*3.*COEF(ID,4))
~~C--------~~
C--------DEFINE LAST BASIS FUNCTION
   20 CONTINUE
                                       ~~GO TO (30,40,50),MODE~~
C        *** MODE=1 *** ADD ILAST-TH BASIS FUNCTION. CONSTRUCT FROM FCT
C                       ILAST-1 BY REFLECTING THE PART OF THE LATTER TO
~~C                       THE RIGHT OF XKNOT ACROSS THE X-AXIS, THEN INTER~~
C                       POLATING. THIS SHOULD INDUCE ONE MORE OSCILLATIO
C                       N IN FCT ILAST THAN IN FCT ILAST-1
~~C~~
   29 MODE = 1
   30 VORDL(1,2) = VORD(ILM1,1,2)
      ~~DO 31 K=1,INSERT~~
      ILOC = IORDER(K)
   31 VORDL(K,1) = VORD(ILM1,ILOC,1)
      ~~DO 32 K=INSERT,INTERV~~
      ILOC = IORDER(K+1)
   32 VORDL(K+1,1) =-VORD(ILM1,ILOC,1)
      ~~VORDL(KNOT,2) =-VORD(ILM1,2,2)~~
                                       GO TO 55
C
~~C        *** MODE=2 *** REPLACE FCT ILAST BY INTERPOLATING IT AT THE~~
C                       CURRENT SET OF KNOTS. IF FCT ILAST HAS NOT BEEN
C                       PREVIOUSLY DEF (INSIRT(ILAST)=0)(SEE 9 ABOVE,
~~C                       ALSO MAIN AT 10)) SET MODE=1,PROCEED IN THAT MOD~~
C
   40 IF (INSIRT(ILAST).EQ.0)          GO TO 29
      ~~VORDL(1,1)=VORD(ILAST,1,1)~~
      VORDL(1,2)=VORD(ILAST,1,2)
      ID = ICLAST
      ~~IBOUND = ICLAST + ILAST - 4~~
      DO 43 K=2,INTERV
   41 IF (ID.EQ.IBOUND)                GO TO 42
      ~~IF (XIL(K).LT.XI(ID+1))           GO TO 42~~
      ID = ID +1
                                       GO TO 41
   ~~42 DX = XIL(K)- XI(ID)~~
   43 VORDL(K,1) = COEF(ID,1)+DX*(COEF(ID,2)+DX*(COEF(ID,3)
     *                                      +DX*COEF(ID,4)))
      VORDL(KNOT,1)=VORD(ILAST,2,1)
      VORDL(KNOT,2)=VORD(ILAST,2,2)
                                       GO TO 55
C
C        *** MODE=3 *** CHANGE FCT ILAST BY CHANGING JUST THE KNOT INTRO
C                       DUCED LAST
C
   50 ID = ICLAST + INSERT - 1
      DX = XKNOT - XI(ID)
      XI(ID) = XKNOT
      IF (DX.GE.0.)                    GO TO 51
      ID = ID - 1
      DX = XKNOT - XI(ID)
   51 VORDL(INSERT,1) = COEF(ID,1) +DX*(COEF(ID,2) +DX*(COEF(ID,3)
     *                                                 +DX*COEF(ID,4)))
C
C        *** INTERPOLATE
   55 CALL INTERP
                                       GO TO (57,57,59),MODE
   57 ID = ICLAST - 1
      DO 56 IO=1,INTERV
      ID = ID + 1
   56 XI(ID) = XIL(IO)
      INSIRT(ILAST) = INSERT
C--------
C--------*** ORTHONORMALIZE FCT ILAST OVER PREVIOUS (ORTHONORMAL) SET
C            THEN COMPUTE THE COMPONENT BC(ILAST) OF UERROR WRTO IT
C            FINALLY,STORE THE VARIOUS REPRESENTATIONS OF FCT ILAST
C
   59 CALL EVAL
      DO 60 I=1,ILM1
   60    TEMP(I) = DOT(I,1)
      DO 69 L=1,LX
   69 FCTL(L) = ARITH1(FCTL(L),ILM1,TEMP,1,FCT(L,1),100)
C  SEE COMMENT IN *DOT* ABOUT THE *ARITH1* ROUTINE
      DO 61 K=1,KNOT
      ILOC = IORDER(K)
      DO 61 L=1,2
   61    VORDL(K,L) = ARITH1(VORDL(K,L),ILM1,TEMP,1,VORD(1,ILOC,L),1)
C  SEE COMMENT IN *DOT* ABOUT *ARITH1* ROUTINE
   67 CALL EVAL
      C = SQRT(DOT(ILAST,1))
      IF (C .EQ. 0.)  C = 1.
      BC(ILAST) = DOT(ILAST,2)  / C
      DO 62 K=1,KNOT
      ILOC = IORDER(K)
      DO 62 L=1,2
      VORDL(K,L) = VORDL(K,L)/C
   62 VORD(ILAST,ILOC,L) = VORDL(K,L)
      ID = ICLAST - 1
      DO 63 IO=1,INTERV
      ID = ID + 1
      DO 63 L=1,4
   63 COEF(ID,L) = COEFL(IO,L)/C
      DO 64 L=1,LX
   64 FCT(L,ILAST) = FCTL(L)/C
C--------
```

```
                                 RETURN                                 SPLIN900
C                                                                       SPLIN901
C     *** THIS OUTPUT INDICATES A FAILURE CONDITION ***                 SPLIN902
   95 WRITE (6,950) XKNOT,ILAST                                         SPLIN903
  950 FORMAT (15H    *** NEW KNOT,E20.8,13H FOR FUNCTION,I3,50H OUT OF BOSPLIN904
     *UNDS OR COINCIDENT WITH A PREVIOUS KNOT./36H   *** EXECUTION CANNOSPLIN905
     *T BE CONTINUED)                                                   SPLIN906
                                 STOP                                   SPLIN907
C                                                                       SPLIN908
      END                                                               SPLIN909
C                                                                       SPLIN910
C***********TREND AND WEIGHT FUNCTIONS**********************************SPLIN911
C                                                                       SPLIN912
```

```
      FUNCTION T(Z)                                                     SPLIN913
      T = 1.                                                            SPLIN914
      RETURN                                                            ~~SPLIN915~~
      END                                                               SPLIN916
                                                                        SPLIN917
      FUNCTION W(Z)                                                     SPLIN918
      W = 1.                                                            SPLIN919
      RETURN                                                            SPLIN920
      END                                                               SPLIN922
C-----------------------------------------------------------------------SPLIN923
```

```
         IDENT    ARITH1
***      ARITH1 - ARITHMETIC PACKAGE FOR LINEQ1 AND DTMNT1.
*
*        DAVID S. DODSON,  06/01/70.


***      FUNCTION.
*
*        GIVEN REAL SCALAR C AND N-VECTORS A AND B, THIS
*        PACKAGE COMPUTES THE FUNCTION:
*                           N
*                          ---
*             ARITH1 = C -  >   A(I) * B(I)
*                          ---
*                          I=1


***      USAGE.
*
*        FORTRAN FUNCTION REFERENCE TO ARITH1 OF FORM:
*
*             Y=ARITH1(C,N,A,KA,B,KB)
*
*        WHERE: A AND B ARE THE NAMES OF THE TWO VECTORS AND
*               KA AND KB ARE THE INCREMENTS BETWEEN SUCCESSIVE
*               ELEMENTS OF THE A AND B VECTORS IN MEMORY.


***      COMPATABILITY.
*
*        THIS ROUTINE IS EQUIVALENT TO THE FORTRAN SUBPROGRAM:
*
*             FUNCTION ARITH1 (C,N,A,KA,B,KB)
*             DOUBLE PRECISION T
*             REAL A(KA,N),B(KB,N)
*             T=DBLE(C)
*             IF(N.EQ.0)GO TO 5
*             DO 4 I=1,N
*           4 T=T-DBLE(A(1,I))*DBLE(B(1,I))
*           5 ARITH1=T
*             RETURN
*             END
         EJECT
         ENTRY    ARITH1


 LOOP    SA1      B3           FETCH NEXT A
         SB3      B3+B4
         SA2      B5           FETCH NEXT B
         SB5      B5+B6
         FX0      X1*X2        (X0,X1) = (X1) * (X2)
         DX1      X1*X2
         FX2      X6-X0        (X6,X7) = (X6,X7) - (X0,X1)
         DX3      X6-X0
         FX0      X7-X1
         NX2      X2
         FX1      X0+X3
         FX0      X1+X2
         NX3      X0
         DX1      X1+X2
         NX2    X1
         FX6    X2+X3
         DX7    X2+X3
         SB2    B2-B1          COUNT TERM
         NZ     B2,LOOP        LOOP TO COMPUTE INNER PRODUCT

ARITH1   BSSZ   1              ENTRY/EXIT
         SA1    B1             (X6,X7) = DBLE(C)
         BX6    X1
         MX7    0
         DX7    X6+X7
         SB1    1              (B1) = 1
         SA1    B2             (B2) = N
         SB2    X1
         ZR     X1,ARITH1      RETURN IF N = 0
         SA1    B4             (B4) = KA
         SB4    X1
         SA1    B6             (B6) = KB
         SB6    X1
         EQ     LOOP
         SPACE  4
         END
```

```
      SUBROUTINE TRID (M,SUP,SUB,DIAG,B)
C
C     ..................................................................
C
C        SUBROUTINE TRID
C
C        DECKS USED
C           TRID
C
C        PURPOSE
C           SOLVES THE MATRIX EQUATION AX=B, WHERE A IS TRIDIAGONAL.
C
C        USAGE
C           CALL TRID(M,SUP,SUB,DIAG,B)
C
C        DESCRIPTION OF PARAMETERS
C              M       - ORDER OF MATRIX A.
C              SUP     - (M X 1) SUPER DIAGONAL OF A.
C                        SUP(I)=A(I,I+1)   I=1,M-1
C              SUB     - (M X 1) SUB DIAGONAL OF A.
C                        SUB(I)=A(I+1,I)   I=1,M-1
C              DIAG    - (M X 1) MAIN DIAGONAL OF A.
C                        DIAG(I)=A(I,I) I=1,M
C              B       - (M X 1) CONSTANT VECTOR. (SOLUTION RETURNED
C                        IN B)
C
C        REMARKS
C           THE ARRAYS MUST HAVE THE FOLLOWING DIMENSIONS
C              SUP(M),SUB(M),DIAG(M),B(M)
C
C           SUB AND SUP CONTAIN M-1 ELEMENTS.
C
C        METHOD
C           DECOMPOSES MATRIX A INTO L*U, THEN SOLVES THE EQUATIONS
C            *Z=B AND U*X=Z.  SOLUTION IS RETURNED IN B VECTOR.
C
C     ..................................................................
C
      DIMENSION SUP(M),SUB(M),DIAG(M),B(M)
      N = M
      NN = N-1
      SUP(1) = SUP(1)/DIAG(1)
      B(1) = B(1)/DIAG(1)
      DO 10 I=2,N
      II = I-1
C     DECOMPOSE A TO FORM A=LU WHERE L IS LOWER TRIANGULAR
C                                    U IS UPPER TRIANGULAR
      DIAG(I) = DIAG(I)-SUP(II)*SUB(II)
      IF (I .EQ. N) GO TO 10
      SUP(I) = SUP(I)/DIAG(I)
C     COMPUTE Z WHERE LZ=B
   10 B(I) = (B(I)-SUB(II)*B(II))/DIAG(I)
C     COMPUTE X BY BACK SUBSITUTION WHERE UX=Z
      DO 20 K=1,NN
      I = N-K
   20 B(I) = B(I)-SUP(I)*B(I+1)
      RETURN
      END
```

APPENDIX III

PROGRAM LISTING - VCF

- ADYNF

- SUBROUTINES

```
      PROGRAM VCF8(INPUT,OUTPUT,PUNCH,TAPE3,TAPE4,
     1TAPE5=INPUT,TAPE6=OUTPUT,TAPE7=PUNCH)
      DIMENSION U(32,2,51),W(32,51).UT(32,51),UB(32,51)
      DIMENSION INFO(16)
      DIMENSION THTASRT(6),UZZT(6),UZZSQT(6),GAMART(6),IFLGT(6)
      DIMENSION THTASRB(6),UZZB(6),UZZSQB(6),GAMARB(6),IFLGB(6)
      DIMENSION CDK(100),CDPK(100),CDSK(100)
      DIMENSION CLSK(100)
      DIMENSION TAWD(53),TAWL(53)
      COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL
      COMMON/KAYS/K,KS,KR,KT,KB,KRT,KRB
      COMMON/ELIPS2/AKPHALF,AKDOTPH
      COMMON/BLOCK1/X(100),Y(100),XB(100),YB(100),XRT(100),YRT(100),
     1XRB(100),YRB(100)
      COMMON/BLOCK2/GAMMA(100),GMAB(100),GMRT(100),GMRB(100)
      COMMON/BLOCK3/XDOT(100),YDOT(100),XDOTB(100),YDOTB(100),
     1XRDOT(100),YRDOT(100),XRDOTB(100),YRDOTB(100)
      COMMON/BLOCK4/TT(100),TB(100),TRT(100),TRB(100)
      COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT
      COMMON/BLOCK10/THETAS,THETASB,THETA
      COMMON/BLOCK11/DS(52),DZ2,DZ8,DZSQ,DZSQ2,DZSQ4,DT2
      COMMON/BLOCK20/DX,DZ,INTX1,NBIG1
      COMMON/BLOCK30/T,TI,DELT,DELTT,DELTB
      COMMON/BLBOX2/TAU,PT4,NBIG
      COMMON/BLBOX12/ZN(51),ISEP
      COMMON/BLBOX13/KTS,IXTRSET,IXBRSET,UTNBIG(53),UBNBIG(53)
      COMMON/BLOCK14/S(53),ST(53),SB(53)
      COMMON/BLBOX14/ZHAT,AKTI
      EXTERNAL CPC,PDRAG,PLIFT
      INTEGER ALPHA
      REAL NOR,NU,MU
      REAL L,LB
      REAL LENGTH
C ..... LENGTH=DIMENSIONAL LENGTH
C ..... AATACK=ANGLE OF ATTACK IN DEGREES
C ..... A FUNCTION SUBPROGRAM NAMED RZERO(ZSTAR) MUST BE
C ..... SUPPLIED BY THE USER TO ENTER THE BODY GEOMETRY.
C ..... ZSTAR IS THE DIMENSIONAL DISTANCE ALONG THE BODY AXIS,
C ..... AND RZERO IS THE CORRESPONDING DIMENSIONAL RADIUS.
C ..... IF THE BODY GEOMETRY IS IN TABULAR FORM THEN RZERO(ZSTAR)
C ..... IS AN APPROXIMATING FUNCTION
C ..... V=FREE STREAM VELOCITY
C ..... NU=KINEMATIC VISCOSITY
C ..... INFO ..... DATA IDENTIFICATION
      READ(5,705)(INFO(I),I=1,16)
      WRITE(6,605)(INFO(I),I=1,16)
      READ(5,706)AATACK,RE,LENGTH
      READ(5,706)DELT,RC,SIGMA
      READ(5,708)KFINAL,TFINAL,ZFINAL
      READ(5,709)LR,LW,LEVEL,KPUN
      PI=4.0*ATAN(1.0)
      DTOR=PI/180.0
      RTOD=180.0/PI
      CALL NONDIM(DMAX,RW,AW,F,SA,LENGTH,PI)
      WRITE(6,611)DMAX,AW,F,SA
      WRITE(6,606)AATACK,RE
      AATACK=DTOR*AATACK
      RE=RE*SIN(AATACK)*AW/LENGTH
      ~~WRITE(6,601)RE,DELT,RC,SIGMA,KFINAL,TFINAL,ZFINAL,LR,LW,LEVEL,KPUN~~
      FTA=F*TAN(AATACK)
C ..... OUTPUT PARAMETERS
      ~~IPUN=0~~
C ..... COUNTERS
      KT=KB=0
      ~~KRT=KRB=0~~
      KR=KS=KSB=1000
      IXTRSET=IXBRSET=INITAL=53
~~C ..... NDIM IS THE K DIMENSION~~
C ..... NDIM MUST BE PUT IN FREEVTX,CPC,AND RVTX
      NDIM=100
C ..... IDIM=DIMENSION OF THETA GRID IN B.L. ROUTINE
C ..... JDIM= DIMENSION OF R GRID IN B.L. ROUTINE
      IDIM=32
      ~~JDIM=51~~
C ..... CONSTANTS
      PI2=2.0*PI
      PINF=132.3/.1189
      SQRTPI=SQRT(PI)
      SQA=SIN(AATACK)**2
      ~~DELT=.125~~
      SRE=SQRT(RE)
C........ ISYM=1 IS THE SYMMETRIC CASE
C........ ISYM=0 IS THE ASYMMETRIC CASE
      ISYM=1
      THTASYM=0.0
~~C********* INITIAL CONDITIONS~~
C********* START OF LOOP
      K=0
      IF(LR.EQ.0) GO TO 219
      READ(LR)  K,KS,KR,KT,KB,KRT,KRB,T,DELT,NBIG,IXTRSET,IXBRSET
       READ(LR)(UTNBIG(I),(UT(I,J),J=1,NBIG),I=1,IXTRSET)
      ~~IF(ISYM.EQ.0) READ(LR)(UBNBIG(I),(UB(I,J),J=1,NBIG),I=1,IXBRSET)~~
      IF(K.LT.KS) GO TO 219
      READ(LR)   (X(I),Y(I),XDOT(I),YDOT(I),GAMMA(I),TT(I),I=1,KT)
      READ(LR)   (XB(I),YB(I),XDOTB(I),YDOTB(I),GMAB(I),TB(I),I=1,KB)
      IF(KRT.EQ.0) GO TO 220
      READ(LR)   (XRT(I),YRT(I),XRDOT(I),YRDOT(I),GMRT(I),TRT(I),I=1,KRT)
      ~~READ(LR)   (XRB(I),YRB(I),XRDOTB(I),YRDOTB(I),GMRB(I),TRB(I),~~
     1I=1,KRB)
  220 CONTINUE
      ~~IF(K.LT.KR) GO TO 219~~
      READ(LR) THTASMT,THTASRT,UZZT,UZZSQT,GAMART,IFLGT,THTASMB,
     1THTASRB,UZZB.UZZSQB,GAMARB,IFLGB,NT,NB,NFT,NFB,GMTLT,GMTLB
  ~~219 CONTINUE~~
      IF((K/KPUN)*KPUN.EQ.K)IPUN=2
      KTS=K+1
      IF(K.NE.0) GO TO 69
      T=TI=(RW/AW)*2.0*FTA*.03
      ZHAT=T*(AW/(RW*2.0*FTA))
      ~~AK=RW/AW*RZRO(ZHAT,LENGTH,RW)~~
      AKTI=AK
      AKDOT=DRZRO(ZHAT,LENGTH,RW)/(2.0*FTA)
      CDPI=PI2*AK*AKDOT
      WRITE(6,620)T,ZHAT,AK,AKDOT,CDPI
   69 K=K+1
      ~~KMIMUS1=K-1~~
      T=T +DELT
      TH=T-DELT/2.0
      ZHAT=T*(AW/(RW*2.0*FTA))
      AK=RW/AW*RZRO(ZHAT,LENGTH,RW)
      AKDOT=DRZRO(ZHAT,LENGTH,RW)/(2.0*FTA)
      AADOT=AK*AKDOT
      AKSQD=AK**2
      ZHALF=TH*(AW/(RW*2.0*FTA))
      AKPHALF=AKHALF=RW/AW*RZRO(ZHALF,LENGTH,RW)
      AKDOTPH=DRZRO(ZHALF,LENGTH,RW)/(2.0*FTA)
C ..... RCMAXSQ AND RCSQ ARE THE MAX CORE RADIUS SQUARED AND CORE
C ..... RADIUS SQUARED
      IF(KS.LE.K)RCMAXSQ=5.04*T/RE
      IF(KS.LE.K)RCMAX=SQRT(RCMAXSQ)
      KTTEMP=KT
      KBTEMP=KB
      KT=KT+1
      KB=KB+1
      CALL BLBOX(SMALLM,DGAMMA,TAWD,TAWL,U,W,UT,UB,IDIM,JDIM,1)
      KT=KTTEMP
      KB=KBTEMP
C ..... SHEAR DRAG
      CDST=0.0
      CLST=0.0
      INTX=ISEP-1
      DO 1 I=1,INTX
      TRAPD=DS(I)/2.0*(TAWD(I)+TAWD(I+1))
      TRAPL=DS(I)/2.0*(TAWL(I)+TAWL(I+1))
      CDST=CDST+TRAPD
 1    CLST=CLST+TRAPL
      CDSK(K)=.50*AK*CDST
      CLSK(K)=.50*AK*CLST
C ..... UPPER HALF OF CYLINDER
 2    IF(K.LT.KS) GO TO 3
      KT=KT+1
      TT(KT)=T
      GAMMA(KT)=DELT*DGAMMA
      GAMMA(KT)=SIGMA*GAMMA(KT)
      X(KT)=(AK+SMALLM)*COS(THETAS)
      Y(KT)=(AK+SMALLM)*SIN(THETAS)
      IF(ISYM.EQ.0) GO TO 3
      KB=KB+1
      XB(KB)=X(KT)
      YB(KB)=-Y(KT)
      GMAB(KB)=-GAMMA(KT)
      TB(KB)=T
 3    CONTINUE
      IF(THETAS.LT.THTASYM.OR.ISYM.EQ.0) GO TO 100
      GO TO 101
C********* SYMMETRIC CASE
 100  KBTEMP=KB
      KB=KB+1
      CALL BLBOX(SMALLM,DGAMMA,TAWD,TAWL,U,W,UT,UB,IDIM,JDIM,2)
      KB=KBTEMP
C ..... ASYMMETRIC CASE
C ..... LOWER HALF OF CYLINDER
C ..... SHEAR DRAG
      CDSB=0.0
      CLSB=0.0
      INTX=ISEP-1
      DO 5 I=1,INTX
      TRAPD=DS(I)/2.0*(TAWD(I)+TAWD(I+1))
      TRAPL=DS(I)/2.0*(TAWL(I)+TAWL(I+1))
      CDSB=CDSB+TRAPD
    5 CLSB=CLSB+TRAPL
      CDSK(K)=CDSK(K)-.50*AK*CDSB
      CLSK(K)=CLSK(K)-.50*AK*CLSB
    6 IF(K.LT.KSB) GO TO 7
      KB=KB+1
      TB(KB)=T
      GMAB(KB)=-DELT*DGAMMA
      GMAP(KB)=SIGMA*GMAB(KB)
      XB(KB)=(AK+SMALLM)*COS(THETASB)
      YB(KB)=(AK+SMALLM)*SIN(THETASB)
    7 CONTINUE
      GO TO 9
  101 CONTINUE
C ..... SYMMETRIC CASE
      CDSK(K)=2.0*CDSK(K)
      CLSK(K)=0.0
      DO 102 I=1,IXTRSET
      UBNBIG(I)=-UTNBIG(I)
      DO 102 J=1,NBIG
  102 UB(I,J)=-UT(I,J)
      IXBRSET=IXTRSET
      THETASB=PI2-THETAS
      KSB=KS
      KB=KT
    9 CONTINUE
C ..... XDOT,YDOT CALCULATIONS
      IF(K.LT.KS) GO TO 85
      CALL VEL(X,Y,XDOT,YDOT,NDIM,1,KT,1)
      IF(ISYM.EQ.0) GO TO 81
      DO 80 I=1,KT
      XDOTB(I)=XDOT(I)
      YDOTB(I)=-YDOT(I)
   80 CONTINUE
      GO TO 82
   81 CONTINUE
      CALL VEL(XB,YB,XDOTB,YDOTB,NDIM,1,KB,2)
   82 CONTINUE
      IF(KRT.EQ.0) GO TO 85
      CALL VEL(XRT,YRT,XRDOT,YRDOT,NDIM,1,KRT,4)
      IF(ISYM.EQ.0) GO TO 84
      DO 83 I=1,KRT
      XRDOTB(I)=XRDOT(I)
      YRDOTB(I)=-YRDOT(I)
   83 CONTINUE
      GO TO 85
   84 CONTINUE
      CALL VEL(XRB,YRB,XRDOTB,YRDOTB,NDIM,1,KRB,5)
   85 CONTINUE
C ..... END XDOT,YDOT CALCULATIONS
C ..... PRINTOUT
C .....
      IPUN=IPUN-1
      ~~IF((K/KPUN)*KPUN.EQ.K) IPUN=2~~
      IW=6
      IF(IPUN.GE.1) IW=7
      ~~IF(K.GE.KS) WRITE(6,600)~~
      IF(K.GE.KS) CALL WRIT(X,Y,XDOT,YDOT,GAMMA,XB,YB,XDOTB,YDOTB,GMAB,
     11,KT,1,KB,K,NDIM,IW)
      ~~IF(KRT.GT.0) WRITE(6,602)~~
      IF(KRT.GT.0) CALL WRIT(XRT,YRT,XRDOT,YRDOT,GMRT,XRB,YRB,XRDOTB,
     *YRDOTB,GMRB,1,KRT,1,KRB,K,NDIM,IW)
~~C ..... END PRINTOUT~~
      CDP=4.0*PI*AKDOT
      SEPDEG=THETAS*RTOD
      ~~IF(K.GE.KS.AND.LEVEL.GE.4)WRITE(6,622)T,ZHAT,AK,AKDOT,SEPDEG~~
      IF(K.GE.KS) CALL DQI(PDRAG,CDP,K,5,ISYM)
      CDPK(K)=.5*AK*CDP
      ~~CDK(K)=CDSK(K)+CDPK(K)~~
      CDN=(AW/RW)*CDK(K)*SQA
      WRITE(6,603)K,CDPK(K),CDSK(K),CDK(K),CDN
      ~~IF(KPUN.NE.0)WRITE(7,755)ZHAT,CDK(K),T,K~~
      IF(K.LT.KR) GO TO 78
      KTEMP=KRT
      ~~CALL RSEP(THETAS,THTASR,ST,UTNBIG,IXTRSET,INITAL,1)~~
      CALL RVTX(XRT,YRT,GMRT,THTASR,THTASMT,THTASRT,UZZT,UZZSQT,GAMART,
     11,KRT,NT,NFT,IFLGT,GMTLT,NDIM,TRT)
      ~~IF(KTEMP.EQ.KRT) GO TO 78~~
      KRB=KRT
      TRB(KRB)=TRT(KRT)
      ~~GMRB(KRB)=-GMRT(KRT)~~
      XRB(KRB)=XRT(KRT)
      YRB(KRB)=-YRT(KRT)
      ~~CALL VEL(XRT,YRT,XRDOT,YRDOT,NDIM,KRT,KRT,4)~~
      XRDOTB(KRB)=XRDOT(KRT)
      YRDOTB(KRB)=-YRDOT(KRT)
      ~~IF(ISYM.EQ.1) GO TO 79~~
      KTEMP=KRB
      CALL RSEP(THETASB,THTASR,SB,UBNBIG,IXBRSET,INITAL,2)
      ~~CALL RVTX(XRB,YRB,GMRB,THTASR,THTASMB,THTASRB,UZZB,UZZSQB,GAMARB,~~
     12,KRB,NB,NFB,IFLGB,GMTLB,NDIM,TRB)
      IF(KTEMP.EQ.KRB) GO TO 78
      ~~CALL VEL(XRB,YRB,XRDOTB,YRDOTB,NDIM,KRB,KRB,5)~~
   79 CONTINUE
   78 CONTINUE
C ..... VORTEX MOTION
      IF(K.LT.KS) GO TO 95
      CALL VM(X,Y,XDOT,YDOT,NDIM,1,KT,DELT)
      ~~IF(ISYM.EQ.0) GO TO 91~~
      DO 90 I=1,KT
      XB(I)=X(I)
      ~~YB(I)=-Y(I)~~
   90 CONTINUE
      GO TO 92
   ~~91 CONTINUE~~
      CALL VM(XB,YB,XDOTB,YDOTB,NDIM,1,KB,DELT)
   92 CONTINUE
      IF(KRT.EQ.0) GO TO 95
      CALL VM(XRT,YRT,XRDOT,YRDOT,NDIM,1,KRT,DELT)
      IF(ISYM.EQ.0) GO TO 94
      ~~DO 93 I=1,KRT~~
      XRB(I)=XRT(I)
      YRB(I)=-YRT(I)
   93 CONTINUE
      GO TO 95
   94 CONTINUE
      CALL VM(XRB,YRB,XRDOTB,YRDOTB.NDIM,1,KRB,DELT)
   95 CONTINUE
C..... END VORTEX MOTION
      CALL SECOND(TIME)
      WRITE(6,612) TIME
      IF(K.EQ.KFINAL.OR.TIME.GE.TFINAL.OR.ZHAT.GE.ZFINAL) GO TO 221
      GO TO 69
  221 CONTINUE
      WRITE(LW) K,KS,KR,KT,KB,KRT,KRB,T,DELT,NBIG,IXTRSET,IXBRSET
      WRITE(LW)(UTNBIG(I),(UT(I,J),J=1,NBIG),I=1,IXTRSET)
      IF(ISYM.EQ.0)WRITE(LW)(UBNBIG(I),(UB(I,J),J=1,NBIG),I=1,IXBRSET)
      IF(K.LT.KS) GO TO 223
      WRITE(LW) (X(I),Y(I),XDOT(I),YDOT(I),GAMMA(I),TT(I),I=1,KT)
      WRITE(LW) (XB(I),YB(I),XDOTB(I),YDOTB(I),GMAB(I),TB(I),I=1,KB)
      IF(KRT.EQ.0) GO TO 222
      WRITE(LW) (XRT(I),YRT(I),XRDOT(I),YRDOT(I),GMRT(I),TRT(I),I=1,KRT)
      WRITE(LW) (XRB(I),YRB(I),XRDOTB(I),YRDOTB(I),GMRB(I),TRB(I),
     1I=1,KRB)
  222 CONTINUE
      IF(K.LT.KR) GO TO 223
      WRITE(LW)THTASMT,THTASRT,UZZT,UZZSQ ,GAMART,IFLGT,THTASMB,
     1THTASRB,UZZB,UZZSQB,GAMARB,IFLGB,NT.NB,NFT.NFB,GMTLT,GMTLB
  223 CONTINUE
  600 FORMAT(//50X*POINT VORTEX LOCATIONS*/27X*TOP BOUNDARY LAYER*44X
     1*BOTTOM BOUNDARY LAYER*//,
     1         *      A     K*6X*X(A)*8X*Y(A)*7X*XDOT(A)*4X*YDOT(A)*6X
     1*GAMMA(A)*3X*XB(A)*7X*YB(A)*6X*XDOTB(A)*4X*YDOTB(A)*6X*GMAB(A)*)
  601 FORMAT(     * 2DUS REYNOLDS NO.=*F11.2,//* 2DUS PARAMETERS*/
     1* DELT=*F5.3,/*    RC=*F5.3,/* SIGMA=*F5.3,///* PROGRAM CONTROL*/
     2* KFINAL=*I3,/* TFINAL=*F6.1,/* ZFINAL=*F5.3,/* LR=*I3,/* LW=*I3,/
     4* LEVEL=*I3,/* KPUN=*I3)
  602 FORMAT(///25X*TOP REAR SHEAR LAYER*44X*BOTTOM REAR SHEAR LAYER*//,
     1         *      A     K*6X*XRT(A)*6X*YRT(A)*5X*XRDOT(A)*3X*YRDOT(A)*5X
     1*GMRT(A)*4X*XRB(A)*5X*YRB(A)*4X*XRDOTB(A)*2X*YRDOTB(A)*4X
     1*GMAB(A)*)
  603 FORMAT(///* K=*I3,/* CDPK=*F12.6,/* CDSK=*F12.6,/*  CDK=*F12.6,
     1/*  CDN=*F12.6)
  605 FORMAT(1H1,20X,16A5 //)
  606 FORMAT(* ANGLE OF ATTACK=*F5.1,* DEGREES*/* 3DS REYNOLDS NO.=*F11.
     12)
  611 FORMAT(//42X*DMAX=*F7.4,/42X*   AW=*F7.4,/42X*    F=*F7.4,/42X,
     1*    S=*F7.4,//)
  612 FORMAT(* ELAPSED TIME=*F12.6)
  620 FORMAT(* ..... TI=*F12.6,* ..... ZHAT(TI)=*F12.6,/* ..... AK(TI)=*
     1F12.6,* ..... AKDOT(TI)=*F12.6,* ..... CDPI=*F12.6)
  622 FORMAT(1H1,40X*PRESSURE DISTRIBUTION*/14X*T=*F7.4,5X*ZHAT=*F7.4,5X
     1*AK=*F7.4,5X*AKDOT=*F7.4,5X*THETAS=*F7.2,//3X*DEG*8X*PHIVT*8X
     2*PHIPT*8X*2(PHIT)*6X*-PSIKSQD*5X*CPK*10X*PDRAG*8X*UTAN*)
  705 FORMAT(16A5)
  706 FORMAT(3F12.6)
  708 FORMAT(I3,9X,2F12.6)
  709 FORMAT(4I2)
  755 FORMAT(3F12.6,I3)
```

```
FUNCTION AN(L,N,IDIM,JDIM,U,W)
DIMENSION U(IDIM,2,JDIM),W(IDIM,JDIM)
COMMON/BLOCK11/DS(52),DZ2,DZ8,DZSQ,DZSQ2.DZSQ4,DT2
AN=DZ8*W(L,N)-DZSQ4
RETURN
ENTRY BN
DX4=.25/DS(L)
BN=DT2+DX4*(U(L+1,2,N)+U(L+1,1,N))+DZSQ2
RETURN
ENTRY CN
CN=-DZ8*W(L,N)-DZSQ4
RETURN
END
```

```
      ~~SUBROUTINE BLBOX(SMALLM,DGAMMA,TAWD,TAWL,U,W,UT,UB,IDIM,JDIM,MODE)~~
      DIMENSION U(IDIM,2,JDIM),W(IDIM,JDIM),UT(IDIM,JDIM),UB(IDIM,JDIM)
      DIMENSION SUP(49),SUB(49),DIAG(49),R(49)
      ~~DIMENSION TAWD(53),TAWL(53)~~
      DIMENSION THTA(53)
      DIMENSION NX(53),DEG(53)
      COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL
      COMMON/KAYS/K,KS,KR,KT,KB,KRT,KRB
      COMMON/ELIPS2/AKPHALF,AKDOTPH
      ~~COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT~~
      COMMON/BLOCK10/THETAS,THETASB,THETA
      COMMON/BLOCK11/DS(52),DZ2,DZ8,DZSQ,DZSQ2.DZSQ4,DT2
      COMMON/BLOCK20/DX,DZ,INTX1,NBIG1
      COMMON/BLOCK30/T,TI,DELT,DELTT,DELTB
      COMMON/BLBOX2/TAU,PT4,NBIG
      ~~COMMON/BLBOX12/ZN(51),ISEP~~
      COMMON/BLBOX13/KTS,IXTRSET,IXBRSET,UTNBIG(53),UBNBIG(53)
      COMMON/BLBOX14/ZHAT,AKTI
      COMMON/BLOCK14/S(53),ST(53),SB(53)
C********* MODE1 IMPLIES 0 DEGREES.LT.THETA.LT. 80 DEGREES
C********* MODE2 IMPLIES -80 DEGREES.LT.THETA.LT. 0 DEGREES
~~C*********RBAR CORRESPONDS TO Z~~
C********* THETA CORRESPONDS TO X
      LEVEL=5
      ~~IF(K.GT.KTS.OR.MODE.EQ.2) GO TO 9~~
C******* INITAL DETERMINES INITAL GRID POINTS
      RTOD=180.0/PI
      ~~DTOR=PI/180.0~~
      INITAL=53
      INTX1=IXTRSET
      ~~IF(IXBRSET.GT.IXTRSET) INTX1=IXBRSET~~
      DELS=5.0*DTOR
      ST(1)=0.0
      ~~SB(1)=PI2~~
      DO 19 I=2,INITAL
      IF(I.LE.15.OR.I.GE.36) GO TO 18
      ~~ST(I)=ST(I-1)+DELS/5.0~~
      SB(I)=SB(I-1)-DELS/5.0
      GO TO 19
  ~~18  ST(I)=ST(I-1)+DELS~~
      SB(I)=SB(I-1)-DELS
   19 CONTINUE
      ~~Z0=0.0~~
      INTZ=50
      INTZ1=INTZ+1
      ~~NBIG1=INTZ~~
      NBIG=INTZ1
      DZ=.14
      ~~DZ2=0.5/DZ~~
      DZ8=0.125/DZ
      DZSQ=1.0/(DZ*DZ)
      ~~DZSQ2=0.5/(DZ*DZ)~~
      DZSQ4=0.25/(DZ*DZ)
      ZN(1)=Z0
~~C*********~~ RBAR VARIATION
      DO 22 J=2,NBIG
      RJ=J
      ~~Z=Z0+(RJ-1.0)*DZ~~
```

```
 22   ZN(J)=Z
      E=10.0**-5
      THETAS=ST(IXTRSET)
      THETASB=SB(IXBRSET)
C********* THETA VARIATION
  9   CONTINUE
C********* TOP HALF
      INTX1=ISEP=IXTRSET
      IF(MODE.EQ.2)INTX1=ISEP=IXBRSET
      INTX=INTX1-1
      IF(K.EQ.1)CALL IC(AKTI,U,IDIM,JDIM,MODE)
      DO 200 I=1,INITAL
      GO TO(204,206) MODE
 204  S(I)=ST(I)
      GO TO 208
 206  S(I)=SB(I)
 208  THTA(I)=S(I)
      DEG(I)=THTA(I)*RTOD
      IF(I.EQ.1) GO TO 213
      DS(I-1)=S(I)-S(I-1)
      DS(I-1)=DS(I-1)*AKHALF**2
C********* B.C. RBAR=0
 213  IF(I.GT.INTX1) GO TO 214
      U(I,1,1)=0.0
      U(I,2,1)=0.0
      JI=2
      GO TO 215
 214  JI=NBIG
 215  DO 200 J=JI,NBIG
C********* B.C. THETA=0
      IF(I.NE.1) GO TO 201
      U(I,1,J)=0.0
      U(I,2,J)=0.0
      GO TO 200
 201  IF(J.NE.NBIG) GO TO 202
C********* B.C. RBAR=INF
      THETA=THTA(I)
      CALL FREEVTX(PSIK1X,PSIK1Y,PSIK1R,3)
      CALL POTFLOW(PSIKXP,PSIKYP,PSIKRP,3)
      IF(I.GT.INTX1) GO TO 203
      U(I,2,NBIG)=AK*(PSIK1R+PSIKRP)
      GO TO (205,207) MODE
 203  GO TO (211,212) MODE
 205  U(I,1,NBIG)=UTNBIG(I)
 211  UTNBIG(I)=AK*(PSIK1R+PSIKRP)
      GO TO 200
 207  U(I,1,NBIG)=UBNBIG(I)
 212  UBNBIG(I)=AK*(PSIK1R+PSIKRP)
      GO TO 200
 202  IF(K.EQ.1) GO TO 200
      GO TO(209,210) MODE
 209  U(I,1,J)=UT(I,J)
      GO TO 200
 210  U(I,1,J)=UB(I,J)
 200  CONTINUE
 40   CONTINUE
      DT2=.5/DELT
      DO 180 I=1,INTX
      NCYCLE=0
      TAUNEW=0.0
      W(I,1)=0.0
   50 NCYCLE=NCYCLE+1
      DO 140 J=2,NBIG1
      TAUS=TAUNEW
      Z=ZN(J)
   90 IF (NCYCLE.GT.1) GO TO 100
      U(I+1,2,J)=UAPRX1(I,J,IDIM,JDIM,U)
  100 DZDX=DZ/(4.0*DS(I))
      W(I,J)=W(I,J-1)-DZDX*(U(I+1,2,J)+U(I+1,2,J-1)+
     1UL(I,J,IDIM,JDIM,U)+UL(I,J-1,IDIM,JDIM,U))
      W(I,J)=W(I,J)-DZ*AKDOTPH/AKPHALF
      IF(J.EQ.2) GO TO 110
      SUB(J-2)=CN(I,J,IDIM,JDIM,U,W)
  110 DIAG(J-1)=BN(I,J,IDIM,JDIM,U,W)
      IF (J.EQ.NBIG1) GO TO 120
      SUP(J-1)=AN(I,J,IDIM,JDIM,U,W)
  120 R(J-1)=DN(I,J,IDIM,JDIM,U,W)
  140 CONTINUE
  160 CALL TRID(49,SUP,SUB.DIAG,R)
      DO 170 JJ=2,NBIG1
      U(I+1,2,JJ)=R(JJ-1)
      IF(MODE.EQ.1) UT(I+1,JJ)=R(JJ-1)
      IF(MODE.EQ.2) UB(I+1,JJ)=R(JJ-1)
  170 CONTINUE
C     ********** WALL FRICTION AT EACH X STATION
      TAUNEW=2.0*U(I+1,2,2)/DZ
      IF(NCYCLE.GE.20) GO TO 179
      IF(ABS(TAUNEW-TAUS).GT.E.OR.NCYCLE.LE.2) GO TO 50
  179  CONTINUE
      NX(I+1)=NCYCLE
  180 CONTINUE
C........ UI GOES TO UBAR
C........  UI REMAINS IN UT AND UB ARRAYS
      DO 182 I=1,INTX1
      DO 182 J=1,NBIG
  182 U(I,2,J)=U(I,2,J)/AK
      IF(MODE.EQ.2) GO TO 28
      IF(LEVEL.GE.4)CALL  RITE(2,IDIM,JDIM,U,ZN,NX,S,DEG,ZHAT,DELS,K)
   28 CONTINUE
C                                .SEARCH ALONG SURFACE FOR ZERO SHEAR POINT
  300 DO 302 I=1,INTX1
      DUDR=(U(I,2,2)-U(I,2,1))/DZ
      TAW=(2.0/SRE)*DUDR
      GO TO(305,306) MODE
  305 IF(I.EQ.1.OR.I.EQ.INITAL.OR.TAW.GT.0.0) GO TO 301
      ISEP=I-1
      GO TO 303
  306 IF(I.EQ.1.OR.I.EQ.INITAL.OR.TAW.LT.0.0) GO TO 301
      ISEP=I-1
      GO TO 303
  301 TAWL(I)=TAW*COS(THTA(I))
  302 TAWD(I)=TAW*SIN(THTA(I))
  303 CONTINUE
C                                .FLOW HAS NOT SEPARATED... SKIP W CALC
      IF(ISEP.EQ.INITAL) GO TO 309
C                                .FLOW HAS SEPARATED ... SEPARATION POINT
```

```
C                                          .DEFINED 5 DEGREES UPSTREAM
  310 IF(MODE.NE.1) GO TO 304
C                                          .TOP
      IXTRSET=ISEP
      THETAS=THTA(ISEP)
      IF(KS.EQ.1000) KS=K
      GO TO 307
C                                          .BOTTOM
  304 IXBRSET=ISEP
      THETASB=THTA(ISEP)
      IF(KSB.EQ.1000) KSB=K
  307 IFLAG=0
      PSIKR=U(ISEP,2,NBIG)
      U00=ABS(PSIKR)
      SMALLM=DELT*U00/PI2
      DGAMMA=(U00**2)/2.0
  309 CONTINUE
      WRITE(6,602)DGAMMA,SMALLM
  602 FORMAT(//* DGAMMA=*F12.6,5X*SMALLM=*F12.6)
      RETURN
      END
      FUNCTION CPC(THTA)
      COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL
      COMMON/KAYS/K,KS,KR,KT,KB,KRT,KRB
      COMMON/PRSURE/XX,YY,SINE,COSINE
      COMMON/BLOCK1/X(100),Y(100),XB(100),YB(100),XRT(100),YRT(100),
     1XRB(100),YRB(100)
      COMMON/BLOCK2/GAMMA(100),GMAB(100),GMRT(100),GMRB(100)
      COMMON/BLOCK3/XDOT(100),YDOT(100),XDOTB(100),YDOTB(100),
     1XRDOT(100),YRDOT(100),XRDOTB(100),YRDOTB(100)
      COMMON/BLOCK4/TT(100),TB(100),TRT(100),TRB(100)
      COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT
      COMMON/BLOCK7/PHIVT,PHIPT,P1,P2,P3,P4
      COMMON/BLOCK10/THETAS,THETASB,THETA
      INTEGER A
      REAL L,LB
      THETA=THTA
      NDIM=100
      PHIVT=0.0
      SINE=SIN(THETA)
      COSINE=COS(THETA)
      IF(KT.LE.1) GO TO 10
      XX=AK*COSINE
      YY=AK*SINE
      KMINUS1=KT-1
      CALL PV(X,Y,XDOT,YDOT,GAMMA,NDIM,1,KMINUS1,SUM,TT)
      PHIVT=PHIVT+SUM
      IF(KRT.EQ.0) GO TO 10
      CALL PV(XRT,YRT,XRDOT,YRDOT,GMRT,NDIM,1,KRT,SUM,TRT)
      PHIVT=PHIVT+SUM
   10 IF(KB.LE.1) GO TO 20
      KMINUS1=KB-1
      CALL PV(XB,YB,XDOTB,YDOTB,GMAB,NDIM,1,KMINUS1,SUM,TB)
      PHIVT=PHIVT+SUM
      IF(KRB.EQ.0) GO TO 20
      CALL PV(XRB,YRB,XRDOTB,YRDOTB,GMRB,NDIM,1,KRB,SUM,TRB)
      PHIVT=PHIVT+SUM
   20 CONTINUE
      PHIPT=2.0*AKDOT*COSINE
      PHIT=PHIPT+PHIVT
      P1=2.0*PHIT
      CALL FREEVTX(PSIK1X,PSIK1Y,PSIK1R,3)
      CALL POTFLOW(PSIKXP,PSIKYP,PSIKRP,3)
      PSIKR2=PSIK1R+PSIKRP
      IF(PSIKR2.LE.-.1.AND.KR.EQ.1000) KR=K
      P4=PSIKR2
      PSIKSQD=PSIKR2**2
      P2=-PSIKSQD
      CPC=2.0*PHIT-PSIKSQD
      P3=CPC
      RETURN
      END
```

```
~~FUNCTION DN(L,N,IDIM,JDIM,U,W)~~
 DIMENSION U(IDIM,2,JDIM),W(IDIM,JDIM)
 COMMON/BLBOX2/TAU,PT4,NBIG
~~COMMON/BLOCK11/DS(52),DZ2,DZ8,DZSQ,DZSQ2,DZSQ4,DT2~~
 COMMON/BLOCK20/DX,DZ,INTX1,NBIG1
 DX8=.125/DS(L)
~~DX2=.5/DS(L)~~
 DN=DX8*U(L+1,2,N)**2-DT2*UM(L,N,IDIM,JDIM,U)-
1DX2*US(L,N,IDIM,JDIM,U)*UL(L,N,IDIM,JDIM,U)
~~DN=DN+DT2*(U(L+1,2,NBIG)+UM(L,NBIG,IDIM,JDIM,U))~~
 DN=DN+DX2*(U(L+1,2,NBIG)/4.0+US(L,NBIG,IDIM,JDIM,U))
1*(U(L+1,2,NBIG)+UL(L,NBIG,IDIM,JDIM,U))
~~UNS1=US(L,N+1,IDIM,JDIM,U)~~
 UNS2=US(L,N+1,IDIM,JDIM,U)-2.0*US(L,N,IDIM,JDIM,U)
 IF(N.EQ.1) GO TO 30
~~UNS1=UNS1-US(L,N-1,IDIM,JDIM,U)~~
 UNS2=UNS2+US(L,N-1,IDIM,JDIM,U)
30 DN=DN-DZ2*UNS1*W(L,N)+DZSQ*UNS2
 IF ~~(N.LT.NBIG1)~~ GO TO 40
 DN=DN-AN(L,NBIG1,IDIM,JDIM,U,W)*U(L+1,2,NBIG)
40 RETURN
~~END~~
```

```
      SUBROUTINE DQI(FCT,CK,K,N,ISYM)                                   DQI
C        . THE FRONT OF THE CYLINDER IS DIVIDED INTO 2N EQUAL PARTS     DQI
C        . THE FRONT IS DEFINED AS -60 DEG.LE.THETA.LE.60 DEG           DQI
C        . THE BACK OF THE CYLINDER IS DIVIDED INTO 8N EQUAL PARTS      DQI
C        . N MUST BE AN EVEN INTEGER                                    DQI
      PI=4.0*ATAN(1.0)                                                  DQI
      IFLAG=0                                                           DQI
      XX=0.0                                                            DQI
      SIMP=0.0                                                          DQI
      H=60.0/N*PI/180.0                                                 DQI
      NB=2*N                                                            DQI
      H3=H/3.0                                                          DQI
      NF=N/2                                                            DQI
      F0=FCT(XX)                                                        DQI
   10 DO 1 I=1,NF                                                       DQI
      XX=XX+H                                                           DQI
      F1=FCT(XX)                                                        DQI
      XX=XX+H                                                           DQI
      F2=FCT(XX)                                                        DQI
      SIMP=SIMP+H3*(F2+4.0*F1+F0)                                       DQI
    1 F0=F2                                                             DQI
      IF(IFLAG.EQ.1) GO TO 5                                            DQI
      H=H/2.0                                                           DQI
      H3=H/3.0                                                          DQI
   20 DO 2 I=1,NB                                                       DQI
      XX=XX+H                                                           DQI
      F1=FCT(XX)                                                        DQI
      XX=XX+H                                                           DQI
      F2=FCT(XX)                                                        DQI
      SIMP=SIMP+H3*(F2+4.0*F1+F0)                                       DQI
    2 F0=F2                                                             DQI
      IF(IFLAG.EQ.0) GO TO 30                                           DQI
      H=H*2.0                                                           DQI
      H3=H/3.0                                                          DQI
      GO TO 10                                                          DQI
   30 KMINUS1=K-1                                                       DQI
      IF(ISYM.EQ.0)GO TO 4                                              DQI
      CK=2.0*SIMP                                                       DQI
      RETURN                                                            DQI
    4 IFLAG=1                                                           DQI
      GO TO 20                                                          DQI
    5 CK=SIMP                                                           DQI
      RETURN                                                            DQI
      END                                                               DQI
      SUBROUTINE FREEVTX(PSIK1X,PSIK1Y,PSIK1R,IA)
      COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL
      COMMON/KAYS/K,KS,KR,KT,KB,KRT,KRB
      ~~COMMON/BLOCK1/X(100),Y(100),XB(100),YB(100),XRT(100),YRT(100),~~
     1XRB(100),YRB(100)
      COMMON/BLOCK2/GAMMA(100),GMAB(100),GMRT(100),GMRB(100)
      ~~COMMON/BLOCK4/TT(100),TB(100),TRT(100),TRB(100)~~
      COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT
      COMMON/BLOCK10/THETAS,THETASB,THETA
      ~~COMMON/VTX1/XX,YY~~
      INTEGER A,B,ALPHA
      REAL LB,L
      ~~NDIM=100~~
      PSIK1R=0.0
      PSIK1X=0.0
      ~~PSIK1Y=0.0~~
      GO TO(1,2,3,4,5) IA
    1 XX=X(ALPHA)
      ~~YY=Y(ALPHA)~~
C ..... DERIVATIVE OF PSI W.R.T. X
C ..... VORTEX MOTION TOP
      ~~GO TO 20~~
    2 XX=XB(ALPHA)
      YY=YB(ALPHA)
~~C ..... DERIVATIVE OF PSI W.R.T. Y~~
C ..... VORTEX MOTION BOTTOM
      GO TO 20
    ~~4 XX=XRT(ALPHA)~~
      YY=YRT(ALPHA)
      GO TO 20
    ~~5 XX=XRB(ALPHA)~~
      YY=YRB(ALPHA)
   20 IF(KT.LE.1) GO TO 23
      ~~KMINUS1=KT-1~~
      CALL PSI(X,Y,GAMMA,NDIM,1,KMINUS1,IA,1,SUM,SUM1,TT)
      PSIK1X=PSIK1X+SUM
      ~~PSIK1Y=PSIK1Y+SUM1~~
      IF(KRT.EQ.0) GO TO 23
      CALL PSI(XRT,YRT,GMRT,NDIM,1,KRT,IA,4,SUM,SUM1,TRT)
      ~~PSIK1X=PSIK1X+SUM~~
      PSIK1Y=PSIK1Y+SUM1
   23 IF(KB.LE.1) RETURN
      ~~KMINUS1=KB-1~~
      CALL PSI(XB,YB,GMAB,NDIM,1,KMINUS1,IA,2,SUM,SUM1,TB)
      PSIK1X=PSIK1X+SUM
      ~~PSIK1Y=PSIK1Y+SUM1~~
      IF(KRB.EQ.0) GO TO 51
      CALL PSI(XRB,YRB,GMRB,NDIM,1,KRB,IA,5,SUM,SUM1,TRB)
      ~~PSIK1X=PSIK1X+SUM~~
      PSIK1Y=PSIK1Y+SUM1
   51 CONTINUF
      ~~RETURN~~
    3 XX=AK*COS(THETA)
      YY=AK*SIN(THETA)
~~C ..... DERIVATIVE OF PSI W.R.T. R ON SURFACE~~
      IF(KT.LE.1) GO TO 34
      KMINUS1=KT-1
      ~~CALL PSIONE(X,Y,GAMMA,NDIM,1,KMINUS1,IA,IB,SUM,SUM1,TT)~~
      PSIK1R=PSIK1R+SUM
      IF(KRT.EQ.0) GO TO 34
      CALL PSIONE(XRT,YRT,GMRT.NDIM,1,KRT,IA,IB,SUM,SUM1,TRT)
      ~~PSIK1R=PSIK1R+SUM~~
   34 IF(KB.LE.1) RETURN
      KMINUS1=KB-1
      CALL PSIONE(XB,YB,GMAB,NDIM,1,KMINUS1,IA,IB,SUM,SUM1,TB)
      PSIK1R=PSIK1R+SUM
      IF(KRB.EQ.0) GO TO 39
      ~~CALL PSIONE(XRB,YRB,GMRB,NDIM,1,KRB,IA,IB,SUM,SUM1,TRB)~~
      PSIK1R=PSIK1R+SUM
   39 CONTINUE
      RETURN
      END
      SUBROUTINE NONDIM(DMAX,RW,AW,F,S,L,PI)
      DIMENSION RZ(20)
      REAL L
      IEND=20
      DZSTAR=L/IEND
      WRITE(6,51)L
   51 FORMAT(////30X*BODY GEOMETRY(DIMENSIONAL LENGTH=*F7.3,*)*//35X,
     1*ZSTAR*11X*RZERO(ZSTAR)*)
      ZSTAR=0.0
      DO 1 I=1,IEND
      RZ(I)=RZERO(ZSTAR)
      WRITE(6,50)ZSTAR,RZ(I)
   50 FORMAT(35XF7.4,13XF7.4)
    1 ZSTAR=ZSTAR+DZSTAR
C ..... SEARCH FOR MAX DIAMETER
      DMAX=0.0
      DO 2 I=2,IEND
      DMAXN=AMAX1(RZ(I-1),RZ(I))
      IF(DMAXN.LE.DMAX) GO TO 2
      DMAX=DMAXN
    2 CONTINUE
      DMAX=2.0*DMAX
      RW=DMAX/2.0
      F=L/DMAX
      S=PI*DMAX**2/4.0
      IF((RZERO(L)-.001).GT.0.0) GO TO 4
      AW=0.0
      DO 3 I=1,IEND
      TRAP=DZSTAR/2.0*(RZ(I)+RZ(I+1))
    3 AW=AW+TRAP
      AW=(1.0/L)*AW
      GO TO 5
    4 AW=DMAX/2.0
    5 CONTINUE
      RETURN
      END
      FUNCTION PDRAG(THETA)
      COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL
      COMMON/BLOCK7/PHIVT,PHIPT,P1,P2,P3,P4
      DEG=THETA*180.0/PI
      CPK=CPC(THETA)
      PDRAG=CPK*COS(THETA)
      WRITE(6,50) DEG,PHIVT,PHIPT,P1,P2,CPK,PDRAG,P4
50    FORMAT(1X,F7.2,7(1X,F12.6))
      RETURN
      ENTRY PLIFT
      CPK=CPC(THETA)
      PLIFT=CPK*SIN(THETA)
      RETURN
      END
      SUBROUTINE POTFLOW(PSIKXP,PSIKYP,PSIKRP,IA)
      COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL
      COMMON/BLOCK1/X(100),Y(100),XB(100),YB(100),XRT(100),YRT(100),
     ~~1XRB(100),YRB(100)~~
      COMMON/BLOCK2/GAMMA(100),GMAB(100),GMRT(100),GMRB(100)
      COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT
      COMMON/BLOCK10/THETAS,THETASB,THETA
      COMMON/BLOCK20/DX,DZ,INTX1,NBIG1
      INTEGER ALPHA,A
      ~~REAL L,LB~~
      GO TO(1,2,3,4,5) IA
 1    L=X(ALPHA)**2+Y(ALPHA)**2
      PSIKXP=2.0*X(ALPHA)*Y(ALPHA)/L**2
      PSIKXP=AKSQD*PSIKXP
      PSIKYP=1.0+AKSQD*(Y(ALPHA)**2-X(ALPHA)**2)/L**2
      ~~RETURN~~
 2    LB=XB(ALPHA)**2+YB(ALPHA)**2
      PSIKXP=2.0*XB(ALPHA)*YB(ALPHA)/LB**2
      ~~PSIKXP=AKSQD*PSIKXP~~
      PSIKYP=1.0+AKSQD*(YB(ALPHA)**2-XB(ALPHA)**2)/LB**2
      RETURN
 3    ~~PSIKRP=2.0*SIN(THETA)~~
      RETURN
 4    L=XRT(ALPHA)**2+YRT(ALPHA)**2
      ~~IF(L.EQ.0.0) RETURN~~
      PSIKXP=2.0*XRT(ALPHA)*YRT(ALPHA)/L**2
      PSIKXP=AKSQD*PSIKXP
      ~~PSIKYP=1.0+AKSQD*(YRT(ALPHA)**2-XRT(ALPHA)**2)/L**2~~
      RETURN
 5    LB=XRB(ALPHA)**2+YRB(ALPHA)**2
      ~~IF(LB.EQ.0.0) RETURN~~
      PSIKXP=2.0*XRB(ALPHA)*YRB(ALPHA)/LB**2
      PSIKXP=AKSQD*PSIKXP
      ~~PSIKYP=1.0+AKSQD*(YRB(ALPHA)**2-XRB(ALPHA)**2)/LB**2~~
      RETURN
      END
      SUBROUTINE PSI(X,Y,GMA,NDIM,KI,KF,IA,IB,SUM,SUM1,TK)
      COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL
      COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT
      COMMON/BLOCK30/T,TI,DELT,DELTT,DELTB
      COMMON/VTX1/XX,YY
      DIMENSION X(NDIM),Y(NDIM),GMA(NDIM),TK(NDIM)
      REAL L
      INTEGER ALPHA,A
      SUM=SUM1=0.0
      DO 1 A=KI,KF
      IF(ALPHA.EQ.A.AND.IA.EQ.IB) GO TO 1
      IFLAG=0
      L=X(A)**2+Y(A)**2
      R1=(XX-X(A))**2+(YY-Y(A))**2
      R2=(XX-X(A)*AKSQD/L)**2+(YY-Y(A)*AKSQD/L)**2
      ARE=XX**2+YY**2
      IF(R1.GT.RCMAXSQ) GO TO 2
      RCSQ=5.04*(T-TK(A))/RE
      IF(R1.GT.RCSQ) GO TO 2
C ..... POTENTIAL VORTEX APPROX INVALID
      IFLAG=1
      GMRSET=GMA(A)
      GMA(A)=0.0
    2 CONTINUE
      SUM=SUM+(GMA(A)/PI2)*((XX-X(A))/R1+XX/ARE-(XX-X(A)*AKSQD/L)/R2)
      SUM1=SUM1+(GMA(A)/PI2)*((YY-Y(A))/R1+YY/ARE-(YY-Y(A)*AKSQD/L)/R2)
      IF(IFLAG.EQ.1)GMA(A)=GMRSET
    1 CONTINUE
      RETURN
      ENTRY PSIONE
      SUM=0.0
      DO 3 A=KI,KF
      IF(X(A).EQ.0.0.AND.Y(A).EQ.0.0) GO TO 3
      IFLAG=0
      L=X(A)**2+Y(A)**2
      R1=(XX-X(A))**2+(YY-Y(A))**2
      RCSQ=RC*RC
      IF(R1.GE.RCSQ) GO TO 4
C ..... POTENTIAL VORTFX APPROX INVALID
      IFLAG=1
      GMRSET=GMA(A)
      GMA(A)=0.0
    4 CONTINUE
      SUM=SUM+(GMA(A)/(PI2*AK))*((AKSQD-L)/R1+1.0)
      IF(IFLAG.EQ.1) GMA(A)=GMRSET
    3 CONTINUE
      RETURN
      END
      ~~SUBROUTINE PV(X,Y,XDT,YDT,GMA,NDIM,KI,KF,SUM,TK)~~
      COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL
      COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT
      ~~COMMON/PRSURE/XX,YY,SINE,COSINE~~
      COMMON/BLOCK30/T,TI,DELT,DELTT,DELTB
      DIMENSION X(NDIM),Y(NDIM),XDT(NDIM),YDT(NDIM),GMA(NDIM)
      ~~DIMENSION TK(NDIM)~~
      REAL L
      INTEGER A
      ~~SUM=0.0~~
      DO 1 A=KI,KF
      L=X(A)**2+Y(A)**2
      ~~IFLAG=0~~
      CAPA=(X(A)*COSINE+Y(A)*SINE)
      R1=AKSQD+L-2.0*AK*CAPA
      ~~R2=(AKSQD*L+AKSQD**2-2.0*AK**3*CAPA)/L~~
      RCSQ=RC*RC
      IF(R1.GE.RCSQ) GO TO 34
      ~~SQR1=SQRT(R1)~~
      WRITE(6,50)A,SQR1
   50 FORMAT(* POTENTIAL VORTEX APPROX INVALID   ALPHA=*I3,5X,
     ~~1*CORE RADIUS=*F12.6)~~
C ..... POTENTIAL VORTEX APPROX INVALID
      IFLAG=1
      ~~GMRSET=GMA(A)~~
      GMA(A)=0.0
   34 CONTINUE
      ~~CAPX=(AKSQD*XDT(A)+2.0*AK*X(A)*AKDOT)/L~~
     1-(2.0*AKSQD*X(A)/L**2)*(X(A)*XDT(A)+Y(A)*YDT(A))
      CAPY=(AKSQD*YDT(A)+2.0*AK*Y(A)*AKDOT)/L
     ~~1-(2.0*AKSQD*Y(A)/L**2)*(X(A)*XDT(A)+Y(A)*YDT(A))~~
      SUM=SUM+(GMA(A)/PI2)*((YDT(A)*(XX-X(A))-XDT(A)*(YY-Y(A)))/R1
     1-(CAPY*(XX-AKSQD*X(A)/L)-CAPX*(YY-AKSQD*Y(A)/L))/R2)
      ~~IF(IFLAG.EQ.1) GMA(A)=GMRSET~~
    1 CONTINUE
      RETURN
      END
      SUBROUTINE RITE(MT,IDIM,JDIM,U,ZN,NX,S,DEG,ZHAT,DELS,K)
      DIMENSION U(IDIM,2,JDIM)
      DIMENSION FOR601(4),FOR560(3),FOR561(3),FOR580(2)
      DIMENSION NX(53),DEG(53)
      DIMENSION S(53),ZN(53)
      COMMON/BLOCK20/DX,DZ,INTX1,NBIG1
      COMMON/BLOCK30/T,TI,DELT,DELTT,DELTB
      COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT
      COMMON/BLBOX2/TAU,PT4.NBIG
      MI=INTX1/12
      I=1
      M=1
      IF(MI.EQ.0) GO TO 35
      MM=12
      IF(MT.EQ.1)WRITE(6,549)TI
      IF(MT.EQ.2)WRITE(6,550) K,T,ZHAT,AK,AKDOT
      WRITE(6,600) (NX(I),I=1,12)
      GO TO 32
   31 WRITE(6,601) (NX(I),I=M,MM)
   32 WRITE(6,560) (S(I),I=M,MM)
      WRITE(6,561) (DEG(I),I=M,MM)
      WRITE(6,580) (ZN(J),(U(I,MT,J),I=M,MM),J=1,NBIG)
      I=I+1
      M=MM+1
      MM=M+11
      IF(I.LE.MI) GO TO 31
      IF(M-1.EQ.INTX1) GO TO 33
   35 MM=INTX1
      N=(MM-M)+1
      NN=N+1
      ENCODE(33,1000,FOR601)N
      ENCODE(29,1004,FOR560) N
      ENCODE(30,1005,FOR561) N
      ENCODE(16,1002,FOR580) NN
      IF(MI.EQ.0)WRITE(6,550)K,T,ZHAT,DELT,DELS,DZ
      WRITE(6,FOR601) (NX(I),I=M,MM)
      WRITE(6,FOR560) (S(I),I=M,MM)
      WRITE(6,FOR561) (DEG(I),I=M,MM)
      WRITE(6,FOR580)(ZN(J),(U(I,MT,J),I=M,MM),J=1,NBIG)
   33 CONTINUE
      RETURN
  549 FORMAT(1H1,40X*BOUNDARY LAYER VELOCITY DISTRIBUTION ( UI )  TI=*
     1F5.3,/)
  550 FORMAT(1H1,50X*BOUNDARY LAYER VELOCITY DISTRIBUTION*/30X* K=*I2,5X
     1*T=*F6.3,5X*ZHAT=*F6.3,5X*AK=*F6.3,5X*AKDOT=*F6.3)
  560 FORMAT(* R↓ (RAD)↑*,1X,12(F9.5,1X))
  561 FORMAT(*    (DEG)↑*,1X,12(F9.5,1X)/)
  580 FORMAT(1X,13(F9.5,1X))
  600 FORMAT(/*   NCYCLE   *,12(4X,I2,4X)//
  601 FORMAT(///*   NCYCLE   *,12(4X,I2,4X)//)
 1000 FORMAT(18H(///*   NCYCLE   *,,I2,13H(4X.I2.4X)//))
 1002 FORMAT(4H(1X,,I2,10H(F9.5,1X)))
 1004 FORMAT(17H(* R↓ (RAD)↑*,1X,,I2,10H(F9.5,1X)))
 1005 FORMAT(17H(*    (DEG)↑*,1X,,I2,11H(F9.5,1X)/))
      END
      SUBROUTINE RSEP(THTASF,THTASR,THTA,UBL,INTX1,INITAL,MODE)
      COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL
      DIMENSION THTA(53),UBL(53)
      ~~UFMX=0.0~~
C ..... SEARCH B.L. FOR MAX VELOCITY
      DO 1 I=2,INTX1
      UFMXN=AMAX1(UBL(I-1),UBL(I))
      IF(UFMXN.LE.UFMX) GO TO 1
      IMAX=I-1
      ~~IF(UFMXN.GT.UBL(I-1)) IMAX=I~~
      UFMX=UFMXN
    1 CONTINUE
      THTAMXF=THTA(IMAX)
      URMX=0.0
C ..... SEARCH S.L. FOR MAX BACKFLOW VELOCITY
      ~~ISLS=INTX1+2~~
      DO 2 I=ISLS,INITAL
      URMXN=AMIN1(UBL(I-1),UBL(I))
      ~~IF(URMXN.GE.URMX)~~ GO TO 2
      JMAX=I-1
      IF(URMXN.LT.UBL(I-1))JMAX=I
      ~~URMX=URMXN~~
    2 CONTINUE
      THTAMXR=THTA(JMAX)
~~C ..... SEARCH S.L. FOR ZERO VELOCITY~~
      ISLE=INITAL-1
      DO 4 I=ISLS,ISLE
      ~~UZN=AMIN1(ABS(UBL(I-1))~~,ABS(UBL(I)))
      IF(I.EQ.ISLS) GO TO 3
      IF(UZN.GE.UZ) GO TO 4
    ~~3 KMAX=I-1~~
      IF(UZN.LT.ABS(UBL(I-1)))KMAX=I
      UZ=UZN
    ~~4 CONTINUE~~
      THTAZRO=THTA(KMAX)
C ..... CALCULATE S.L. SEPARATION POINT
      ~~X=(THTAZRO-THTAMXF)/(THTASF-THTAMXF)~~
      THTASR=THTAMXR-((THTAMXR-THTAZRO)/X)
      THTAMXF=THTAMXF*RTOD
      ~~THTAMXR=THTAMXR*RTOD~~
      THTAZRO=THTAZRO*RTOD
      THTASRD=THTASR*RTOD
      ~~WRITE(6,50)UFMX,THTAMXF,URMX,THTAMXR,UZ,THTAZRO,THTASRD~~
   50 FORMAT(1H1,* MAX VELOCITY=*F6.3,* AT THETA=*F5.1,* DEGREES*/
     1* MAX BACKFLOW VEL=*F6.3,* AT THETA=*F5.1,* DEGREES*/
     ~~2* MIN TANGENTIAL VELOCITY BETWEEN MAX VELOCITY AND MAX BACKFLOW VE~~
     3LOCITY=*F6.3,* AT THETA=*F5.1,* DEGREES*/* REAR SEPARATION ANGLE=*
     4F5.3,* DEGREES*)
      ~~RETURN~~
      END
      SUBROUTINE RVTX(X,Y,GMA,THTASR,THTASM,THTAS,UZZ,UZZSQ,GAMA,
     1MODE,KRN,N,NF,IFLAG,GMATL,NDIM,TK)
      DIMENSION X(NDIM),Y(NDIM),GMA(NDIM),TK(NDIM)
      ~~DIMENSION THTAS(6),UZZ(6),UZZSQ(6),GAMA(6),IFLAG(6)~~
      COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL
      COMMON/KAYS/K,KS,KR,KT,KB,KRT,KRB
      COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT
      COMMON/BLOCK10/THETAS,THETASB,THETA
      COMMON/BLOCK30/T,TI,DELT,DELTT,DELTB
      ~~IF(K.NE.KR) GO TO 1~~
      NF=5
      N=NF
      GMATL=0.
1     N=N+1
      THTAS(N)=THTASR
      ~~THETA=THTAS(N)~~
      CALL FREEVTX(PSIK1X,PSIK1Y,PSIK1R,3)
      CALL POTFLOW(PSIKXP,PSIKYP,PSIKRP,3)
      UZZ(N)=PSIK1R+PSIKRP
      THTAP=THTAS(N)*RTOD
      WRITE(6,600)N,THTAP,UZZ(N)
~~600   FORMAT(//* RVTX SUBROUTINE N=*I3,2X*THETAS(N)=*F12.6,~~
     12X*U00(N)=*F12.6)
      UZZ(N)=ABS(UZZ(N))
      UZZSQ(N)=UZZ(N)*UZZ(N)
      GAMA(N)=-DELT*UZZSQ(N)/2.0
      IF(MODE.EQ.2)GAMA(N)=-GAMA(N)
      ~~WRITE(6,601)GAMA(N)~~
601   FORMAT(* GAMA(N)=*F12.6)
      IF(N.EQ.NF+1) GO TO 10
      ~~IFLAG(N-1)=0~~
      IF(THTAS(N)-THTAS(N-1))3,4,2
2     IF(MODE.EQ.1) IFLAG(N-1)=1
      ~~GO TO 5~~
3     IF(MODE.EQ.2) IFLAG(N-1)=1
      GO TO 5
~~4     IFLAG(N-1)=1~~
5     CONTINUE
      IF(IFLAG(N-1).EQ.1)GMATL=GMATL+GAMA(N-1)
      ~~WRITE(6,602)GMATL,N,IFLAG(N-1)~~
602   FORMAT(/* GAMA CHECK SUM=*F12.6,5X*N=*I3,5X*IFLAG(N-1)=*I3)
      AGMATL=ABS(GMATL)
      ~~IF(AGMATL.LE..1) GO TO 6~~
      IF(GMATL.LE..1) GO TO 6
      NF=N
      ~~GO TO 7~~
6     IF(N.LT.5) RETURN
7     THTASM=THTAS(N)
      ~~GMATL=0.~~
      RETURN
10    IF(K.EQ.KR) GO TO 2000
      ~~IFLAG(NF)=0~~
      IF(THTAS(N)-THTASM)30,40,20
20    IF(MODE.EQ.1) IFLAG(NF)=1
      ~~GO TO 50~~
30    IF(MODE.EQ.2) IFLAG(NF)=1
      GO TO 50
~~40    IFLAG(NF)=1~~
```

```
 50   CONTINUE
      THTAP=THTAS(N)*RTOD
      THTAPM=THTASM*RTOD
      WRITE(6,603)N,NF,IFLAG(NF),THTAP,THTAPM
 603  FORMAT(/* OVERLAP CHECK N=*I3,5X*NF=*I3,5X*IFLAG(NF)=*I3,
     15X*THTAS(N)=*F12.6,5X*THTASM=*F12.6)
      U00=0.0
      U00SQ=0.0
      GAM=0.
      THTASR=0.
      RN=0.
      N=1
 100  IF(IFLAG(N).EQ.0) GO TO 200
      RN=RN+1.0
      U00=U00+UZZ(N)
      U00SQ=U00SQ+UZZSQ(N)
      GAM=GAM+GAMA(N)
      THTASR=THTASR+THTAS(N)
      WRITE(6,604)N,U00,U00SQ,GAM,THTASR
 604  FORMAT(/* LUMP SUMS N=*I3,2X*U00=*F12.6,2X*U00SQ=*F12.6,
     12X*GAM=*F12.6,2X*THTASR=*F12.6)
 200  IF(N.EQ.NF) GO TO 1000
      N=N+1
      GO TO 100
1000  CONTINUE
      SMALLM=(RN*DELT/PI2)*(U00SQ/U00)
      THTASR=THTASR/RN
      KRN=KRN+1
      X(KRN)=(AK+SMALLM)*COS(THTASR)
      Y(KRN)=(AK+SMALLM)*SIN(THTASR)
      GMA(KRN)=GAM
      GMA(KRN)=SIGMA*GMA(KRN)
      THTAP=THTASR*RTOD
      TK(KRN)=T
      WRITE(6,605)SMALLM,THTAP,KRN,X(KRN),Y(KRN),GMA(KRN)
 605  FORMAT(//* RVTX BIRTH   SMALLM=*F12.6,5X*THTASR(AVERAGE)=*F12.6,/
     15X*KRN=*I3,2X*X(KRN)=*F12.6,2X*Y(KRN)=*F12.6,2X*GMA(KRN)=*F12.6)
2000  UZZ(1)=UZZ(NF+1)
      UZZSQ(1)=UZZSQ(NF+1)
      GAMA(1)=GAMA(NF+1)
      THTAS(1)=THTAS(NF+1)
      N=1
      NF=5
      RETURN
      END
```

```
FUNCTION RZRO(ZHAT,L,RW)
REAL L
RZRO=RZERO(ZHAT*L)/RW
RETURN
ENTRY DRZRO
RZRO=(L/RW)*DRZERO(ZHAT*L)
RETURN
END
```

```
      FUNCTION UAPRX1(L,N,IDIM,JDIM,U)                                  UAPRX1 1
      DIMENSION U(IDIM,2,JDIM)                                          UAPRX1 2
C     **********FIRST APPROXIMATION TO U(L+1,2,N)                       UAPRX1 3
      UAPRX1=U(L+1,1,N)+U(L,2,N)-U(L,1,N)                               UAPRX1 4
      RETURN                                                            UAPRX1 5
      ENTRY US                                                          UAPRX1 6
      US= (U(L+1,1,N)+U(L,2,N)+U(L,1,N))/4.0                            UAPRX1 7
      RETURN                                                            UAPRX1 8
      ENTRY UL                                                          UAPRX1 9
      UL=U(L+1,1,N)-U(L,2,N)-U(L,1,N)                                   UAPRX 10
      RETURN                                                            UAPRX 11
      ENTRY UM                                                          UAPRX 12
      UM= U(L,2,N)-U(L+1,1,N)-U(L,1,N)                                  UAPRX 13
      RETURN                                                            UAPRX 14
      END                                                               UAPRX 15
      SUBROUTINE VEL(X,Y,XDT,YDT,NDIM,KI,KF,IA)
      COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL
      COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT
      DIMENSION X(NDIM),Y(NDIM),XDT(NDIM),YDT(NDIM)
      INTEGER ALPHA
      REAL L
      DO 1 ALPHA=KI,KF
      L=X(ALPHA)**2+Y(ALPHA)**2
      CALL FREEVTX(PSIK1X,PSIK1Y,PSIK1R,IA)
      CALL POTFLOW(PSIKXP,PSIKYP,PSIKRP,IA)
      PSIKX=PSIKXP+PSIK1X+AK*AKDOT*Y(ALPHA)/L
      PSIKY=PSIKYP+PSIK1Y-AK*AKDOT*X(ALPHA)/L
      XDT(ALPHA)=-PSIKY
      YDT(ALPHA)=PSIKX
    1 CONTINUE
      RETURN
      END
      SUBROUTINE VM(X,Y,XDT,YDT,NDIM,KI,KF,DT)
      COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL
      COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT
      DIMENSION X(NDIM),Y(NDIM),XDT(NDIM),YDT(NDIM)
      INTEGER ALPHA
      DO 1 ALPHA=KI,KF
      XTEMP=X(ALPHA)
      YTEMP=Y(ALPHA)
      X(ALPHA)=X(ALPHA)+DT*XDT(ALPHA)
      Y(ALPHA)=Y(ALPHA)+DT*YDT(ALPHA)
      R=SQRT(X(ALPHA)**2+Y(ALPHA)**2)
      IF(R.GT.AK) GO TO 1
      WRITE(6,600) X(ALPHA),Y(ALPHA)
      CALL VMFIX(XTEMP,YTEMP,XDT(ALPHA),YDT(ALPHA))
      X(ALPHA)=XTEMP+DT*XDT(ALPHA)
      Y(ALPHA)=YTEMP+DT*YDT(ALPHA)
      WRITE(6,601) X(ALPHA),Y(ALPHA)
1     CONTINUE
600   FORMAT(//* BEFORE VMFIX(X(ALPHA),Y(ALPHA))=(*F8.5,*,*F8.5,*)*)
601   FORMAT(//* AFTER  VMFIX(X(ALPHA),Y(ALPHA))=(*F8.5,*,*F8.5,*)*)
      RETURN
      END
      SUBROUTINE VMFIX(X,Y,XDOT,YDOT)
      COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL
      COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT
      COMMON/BLOCK10/THETAS,THETASB,THETA
      DEF=.001
      IF(X.NE.0) GO TO 10
      L=1
      ANGLE=PI/2.0
      IF(Y.LT.0)L=2
      IF(Y.LT.0) ANGLE=(3.0/2.0)*PI
      GO TO 30
   10 IF(Y.NE.0) GO TO 20
      L=1
      ANGLE=0.0
      IF(X.LT.0)L=2
      IF(X.LT.0) ANGLE=PI
      GO TO 30
   20 IF(X.GT.0.0.AND.Y.GT.0.0)L=1
      IF(X.LT.0.0.AND.Y.GT.0.0)L=2
      IF(X.LT.0.0.AND.Y.LT.0.0)L=3
      IF(X.GT.0.0.AND.Y.LT.0.0)L=4
      ANGLE=ATAN(Y/X)
   30 CONTINUE
      SINE=SIN(ANGLE)
      COSINE=COS(ANGLE)
      GO TO(1,2,3,4),L
    1 X=(AK+DEF)*COSINE
      Y=(AK+DEF)*SINE
      XDOT=XDOT*SINE
      YDOT=YDOT*COSINE
      U=-XDOT+YDOT
      XDOT=-U*SINE
      YDOT=U*COSINE
      GO TO 5
    2 X=-(AK+DEF)*COSINE
      Y=-(AK+DEF)*SINE
      XDOT=-XDOT*SINE
      YDOT=YDOT*COSINE
      U=-XDOT-YDOT
      XDOT=U*SINE
      YDOT=-U*COSINE
      GO TO 5
    3 X=-(AK+DEF)*COSINE
      Y=-(AK+DEF)*SINE
      XDOT=XDOT*SINE
      YDOT=YDOT*COSINE
      U=XDOT-YDOT
      XDOT=U*SINE
      YDOT=-U*COSINE
      GO TO 5
    4 X=(AK+DEF)*COSINE
      Y=(AK+DEF)*SINE
      XDOT=-XDOT*SINE
      YDOT=YDOT*COSINF
      U=XDOT+YDOT
      XDOT=-U*SINE
      YDOT=U*COSINE
      GO TO 5
```

```
 5    CONTINUE
      RTOD=180.0/PI
      ADEG=ANGLE*RTOD
```
~~WRITE(6,50)AL³HA,ADEG,X,Y,XDOT,YDOT~~
```
 50   FORMAT(//* VMFIX*2X*ALPHA=*I3,2X*ANGLE=*F12.6,/*  X=*F12.6,
     12X*Y=*F12.6,2X*XDOT=*F12.6,2X*YDOT=*F12.6)
      RETURN
      END
      SUBROUTINE WRIT(X,Y,XDOT,YDOT,GAMMA,XB,YB,XDOTB,YDOTB,GMAB,
     1KIT,KFT,KIB,KFB,K,NDIM,IW)
      DIMENSION X(NDIM),Y(NDIM),XDOT(NDIM),YDOT(NDIM),GAMMA(NDIM),
     1XB(NDIM),YB(NDIM),XDOTB(NDIM),YDOTB(NDIM),GMAB(NDIM)
      IF(KFT-KFB) 1,2,2
    1 KF=KFB
      GO TO 3
    2 KF=KFT
    3 CONTINUE
      IF(KIT.GT.KIB) GO TO 17
      DO 16 I=KIT,KF
      IF(I.GE.KSB) GO TO 12
   11 WRITE(6,601)I,K,X(I),Y(I),XDOT(I),YDOT(I),GAMMA(I)
      IF(IW.NE.7.AND.IW.NE.9) GO TO 16
      WRITE(IW,701) I,K,X(I),Y(I),XDOT(I),YDOT(I),GAMMA(I)
      GO TO 16
   12 IF(KFT.NE.KFB) GO TO 14
   13 WRITE(6,603) I,K,X(I),Y(I),XDOT(I),YDOT(I),GAMMA(I),
     1XB(I),YB(I),XDOTB(I),YDOTB(I),GMAB(I)
      IF(IW.NE.7.AND.IW.NE.9) GO TO 16
      WRITE(IW,703) I,K,X(I),Y(I),XDOT(I),YDOT(I),GAMMA(I),XB(I),YB(I),
     1XDOTB(I),YDOTB(I),GMAB(I)
      GO TO 16
   14 IF(KFT.LT.KFB) GO TO 15
      IF(I.LE.KFB) GO TO 13
      GO TO 11
   15 IF(I.LE.KFT) GO TO 13
   16 CONTINUE
      GO TO 20
   17 CONTINUE
      DO 19 I=KIB,KF
      IF(I.GE.KS) GO TO 18
      WRITE(6,602) I,K,XB(I),YB(I),XDOTB(I),YDOTB(I),GMAB(I)
      IF(IW.NE.7.AND.IW.NE.9) GO TO 19
      WRITE(IW,702) I,K,XB(I),YB(I),XDOTB(I),YDOTB(I),GMAB(I)
      GO TO 19
   18 WRITE(6,603) I,K,X(I),Y(I),XDOT(I),YDOT(I),GAMMA(I),
     1XB(I),YB(I),XDOTB(I),YDOTB(I),GMAB(I)
      IF(IW.NE.7.AND.IW.NE.9) GO TO 19
      WRITE(IW,703) I,K,X(I),Y(I),XDOT(I),YDOT(I),GAMMA(I),XB(I),YB(I),
     1XDOTB(I),YDOTB(I),GMAB(I)
   19 CONTINUE
   20 CONTINUE
  601 FORMAT(2XI3,2XI3,5(2X,F10.6))
  602 FORMAT(2XI3,2XI3,60X,5(2X,F10.6))
  603 FORMAT(2XI3,2XI3,10(2X,F10.6))
  701 FORMAT(2I4,5F7.3)
  702 FORMAT(2I4,35X,4F7.3)
  703 FORMAT(2I4,10F7.3)
      RETURN
      END
      PROGRAM ADYNF(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE
      COMMON/OUTPUT/UERROR(100),FCTL(100),XIL(28),COEFL(27,4),
     *              VORDL(28,2),KNOT,LMAX,INTERV
      COMMON/OTHER/LXI,LXI1,LXI2,Q,CHANGE,ERROR,ACC,XI(28)
      DIMENSION INFO(16),X(100),Z(100),PORCD(100),CDN(100)
      DIMENSION Y(100)
      REAL L
      REAL LENGTH,LAMBDA
      EXTERNAL FIT
C ..... INFO ..... DATA IDENTIFICATION
      READ(5,605) (INFO(I),I=1,16)
  605 FORMAT(16A5)
      WRITE(6,651)(INFO(I),I=1,16)
  651 FORMAT(1H1,20X,16A5 //)
      READ(5,652)AATACK,LAMBDA,LENGTH,F,AW,RW
  652 FORMAT(6F12.6)
C ..... READ NO. OF KNOTS, NO. OF POINTS, X CO-ORD, Y CO-ORD
      READ(5,610)NOKNOT,LX,(X(I),PORCD(I),I=1,LX)
  610 FORMAT(2I4,/(2F12.8))
C ..... IF NOKNOT .GT.0 READ IN KNOT POSITIONS
      LXI2=IABS(NOKNOT)
      IF(NOKNOT.GT.0)READ(5,601) (XI(J),J=1,LXI2)
  601 FORMAT(6F12.6)
C ..... ELLIPSOID PARAMETERS
      PI=4.0*ATAN(1.0)
      DTOR=PI/180.0
      RTOD=180.0/PI
      AATACK=AATACK*DTOR
      FTA=F*TAN(AATACK)
      COEFN=4.0*F/PI
      COEFM=-4.0*F/PI
      COEF1=AW/RW*SIN(AATACK)**2
C ..... CALCULATE CDN,ZHAT
C ..... GET SPLINE X CO ORD  XX(I), Y CO ORD U(I)
      DO 1 I=1,LX
      Z(I)=X(I)
      CDN(I)=COEF1*PORCD(I)
      U(I)=CDN(I)
 1    XX(I)=Z(I)
      CALL SPLINEB(NOKNOT)
C ..... TEMPORARY DEBUGGING DATA PLOT Y VS. X
      WRITE(6,62)
      DO 3 I=1,21
      X(I)=.05*(I-1)
      Y(I)=FIT(X(I))
 3    WRITE(6,61)X(I),Y(I)
 61   FORMAT(F12.6,4X,F12.6)
 62   FORMAT(1H1,6X*Z*13X*CDN(Z)*)
      CNI=CADRE(FIT,0.0,1.0,.01,.01,3,ERR,IFLAG)
      CN=COEFN*CNI
      WRITE(6,60)CN,CNI,ERR,IFLAG
 60   FORMAT(/////* ..... CN=*F12.6,/* ..... CNI=*F12.6,5X*ERROR=*F12.8,
     15X*IFLAG=*I2)
      DO 2 I=1,LX
      CDN(I)=(Z(I)+RZRO(Z(I),LENGTH,RW)*DRZRO(Z(I),LENGTH,RW)/
     1(4.0*F**2))*CDN(I)
      XX(I)=Z(I)
  2   U(I)=CDN(I)
      CALL SPLINEB(NOKNOT)
C ..... TEMP DEBUG DATA SPLINE Y VS X
      WRITE(6,63)
      DO 4 I=1,21
      X(I)=.05*(I-1)
      Y(I)=FIT(X(I))
  4   WRITE(6,61)X(I),Y(I)
 63   FORMAT(1H1,6X*Z*13X*(.5-Z)CDN*)
      CMI=CADRE(FIT,0.0,1.0,.01,.01,3,ERR,IFLAG)
      CM=COEFM*CMI+LAMBDA*CN
      WRITE(6,65)CM,CMI,ERR,IFLAG
 65   FORMAT(/////* ..... CM=*F12.6,/* ..... CMI=*F12.6,5X*ERROR=*F12.8,
     15X*IFLAG=*I2)
      END
      FUNCTION FIT(X)
      COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE
      COMMON/OUTPUT/UERROR(100),FCIL(100),XIL(28),COEFL(27,4),
     *               VORDL(28,2),KNOT,LMAX,INTERV
      COMMON/OTHER/LXI,LXI1,LXI2,D,CHANGE,ERROR,ACC,XI(28)
      I=LXI2
    1 A=X-XI(I)
      IF(A)2,2,4
    2 I=I-1
      IF(I)3,3,1
    3 I=1
    4 FIT=COEFL(I,1)+A*(COEFL(I,2)+A*(COEFL(I,3)+A*COEFL(I,4)))
      RETURN
      END
```

APPENDIX IV

SAMPLE CASE

SAMPLE

FUNCTION RZERO

(INPUT BODY GEOMETRY)

```
      FUNCTION RZERO(ZSTAR)
      REAL LN
      D=4.7
      LN=14.1
      R=43.475
      IF(ZSTAR.GE.LN) GO TO 1
      RZERO=SQRT(R**2-(ZSTAR-LN)**2)-(R-D/2.0)
      GO TO 2
1     RZERO=D/2.0
2     CONTINUE
      RETURN
      ENTRY DRZERO
      IF(ZSTAR.GE.LN) GO TO 3
      RZERO=(LN-ZSTAR)/SQRT(R**2-(ZSTAR-LN)**2)
      GO TO 4
3     RZERO=0.0
4     CONTINUE
      RETURN
      END
```

```
          VCF      SAMPLE      INPUT
          CASE ... OGIVE CYLINDER ... ANGLE OF ATTACK (15 DEG)


          *****     VCF   SAMPLE   OUTPUT      *****                   CARD 1
15.0        4725600.0   50.478                                          CARD 2
.125        .05         .6                                              CARD 3
 39         90.0        1.0                                             CARD 4
 3 4 5 5                                                                CARD 5
```

```
                ***** -VCF- SAMPLE OUTPUT  *****



                BODY GEOMETRY(DIMENSIONAL LENGTH= 50.478)

                    ZSTAR              RZERO(ZSTAR)
                     0.0000                 0.0000
                     2.5239                  .7805
                     5.0478                 1.3972
                     7.5717                 1.8571
                    10.0956                 2.1652
                    12.6195                 2.3248
                    15.1434                 2.3500
                    17.6673                 2.3500
                    20.1912                 2.3500
                    22.7151                 2.3500
                    25.2390                 2.3500
                    27.7629                 2.3500
                    30.2868                 2.3500
                    32.8107                 2.3500
                    35.3346                 2.3500
                    37.8585                 2.3500
                    40.3824                 2.3500
                    42.9063                 2.3500
                    45.4302                 2.3500
                    47.9541                 2.3500

                           RMAX= 4.7000
                             AW= 2.3500
                              F=10.7400
                              S=17.3494

ANGLE OF ATTACK= 15.0 DEGREES
3DS REYNOLDS NO.= 4725600.00
2DLS REYNOLDS NO.=   56940.19

2DLS PARAMETERS
DELT= .125
  RC= .050
SIGMA= .600


PROGRAM CONTROL
KFINAL= 39
TFINAL= 90.0
ZFINAL=1.000
LR=  3
LW=  4
LEVEL= 5
KPLN=  5
```

```
                                              BOUNDARY LAYER VELOCITY DISTRIBUTION
                             K=39      T= 5.048       ZHAT= 2.686      AK= 1.000      AKDOT= 0.000

  NCYCLE          0          3          3          3          3          3          3          4          4          4          4          4

 R+ (RAD)+   0.00000     .08727     .17453     .26180     .34907     .43633     .52360     .61087     .69813     .78540     .87266     .95993
    (DEG)+   0.00000    5.00000   10.00000   15.00000   20.00000   25.00000   30.00000   35.00000   40.00000   45.00000   50.00000   55.00000

   0.00000   0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
    .14000   0.00000     .03785     .07501     .11081     .14459     .17570     .20353     .22752     .24710     .26176     .27101     .27436
    .28000   0.00000     .06936     .13755     .20341     .26581     .32363     .37579     .42128     .45910     .48829     .50792     .51704
    .42000   0.00000     .09499     .18847     .27901     .36514     .44543     .51848     .58293     .63745     .68074     .71147     .72627
    .56000   0.00000     .11531     .22896     .33929     .44466     .54346     .63407     .71492     .78442     .84101     .88307     .90687
    .70000   0.00000     .13105     .26037     .38621     .50684     .62055     .72562     .82033     .90296     .97175    1.02486    1.06030
    .84000   0.00000     .14293     .28415     .42185     .55430     .67975     .79644     .90261     .99645    1.07609    1.13957    1.18472
    .98000   0.00000     .15171     .30170     .44824     .58962     .72410     .84993     .96533    1.06850    1.15752    1.23037    1.28478
   1.12000   0.00000     .15799     .31431     .46729     .61524     .75649     .88931    1.01199    1.12271    1.21960    1.30063    1.36351
   1.26000   0.00000     .16237     .32313     .48066     .63333     .77953     .91758    1.04583    1.16251    1.26582    1.35375    1.42408
   1.40000   0.00000     .16534     .32913     .48979     .64576     .79548     .93734    1.06974    1.19101    1.29938    1.39296    1.46961
   1.54000   0.00000     .16730     .33309     .49585     .65406     .80621     .95078    1.08620    1.21088    1.32315    1.42122    1.50304
   1.68000   0.00000     .16855     .33563     .49975     .65944     .81324     .95966    1.09721    1.22438    1.33957    1.44109    1.52702
   1.82000   0.00000     .16933     .33720     .50218     .66283     .81770     .96536    1.10438    1.23330    1.35060    1.45469    1.54380
   1.96000   0.00000     .16979     .33815     .50366     .66489     .82045     .96892    1.10892    1.23903    1.35782    1.46378    1.55526
   2.10000   0.00000     .17006     .33870     .50452     .66611     .82209     .97107    1.11170    1.24261    1.36242    1.46970    1.56289
   2.24000   0.00000     .17021     .33902     .50501     .66681     .82304     .97234    1.11336    1.24479    1.36527    1.47345    1.56785
   2.38000   0.00000     .17029     .33918     .50529     .66720     .82357     .97305    1.11432    1.24607    1.36699    1.47576    1.57099
   2.52000   0.00000     .17034     .33927     .50542     .66740     .82386     .97345    1.11486    1.24680    1.36799    1.47714    1.57292
   2.66000   0.00000     .17036     .33932     .50549     .66751     .82401     .97366    1.11515    1.24721    1.36856    1.47795    1.57408
   2.80000   0.00000     .17037     .33934     .50553     .66756     .82408     .97376    1.11530    1.24742    1.36888    1.47841    1.57476
   2.94000   0.00000     .17037     .33935     .50555     .66759     .82412     .97382    1.11538    1.24754    1.36904    1.47867    1.57515
   3.08000   0.00000     .17037     .33936     .50555     .66760     .82414     .97384    1.11542    1.24760    1.36913    1.47880    1.57537
   3.22000   0.00000     .17038     .33936     .50556     .66760     .82415     .97386    1.11544    1.24762    1.36918    1.47887    1.57548
   3.36000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24764    1.36920    1.47891    1.57554
   3.50000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24764    1.36921    1.47892    1.57557
   3.64000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36921    1.47893    1.57559
   3.78000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36921    1.47894    1.57560
   3.92000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   4.06000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   4.20000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   4.34000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   4.48000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   4.62000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   4.76000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   4.90000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   5.04000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   5.18000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   5.32000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   5.46000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   5.60000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   5.74000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   5.88000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   6.02000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   6.16000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   6.30000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   6.44000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   6.58000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   6.72000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   6.86000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
   7.00000   0.00000     .17038     .33936     .50556     .66761     .82415     .97386    1.11545    1.24765    1.36922    1.47894    1.57560
```

NCYCLE	4	5	5	5	5	5	6	6	6	6	6	6
R* (RAD)*	1.04720	1.13446	1.22173	1.23918	1.25664	1.27409	1.29154	1.30900	1.32645	1.34390	1.36136	1.37881
(DEG)*	60.00000	65.00000	70.00000	71.00000	72.00000	73.00000	74.00000	75.00000	76.00000	77.00000	78.00000	79.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
.14000	.27127	.26105	.24270	.23763	.23241	.22661	.22046	.21372	.20643	.19848	.18983	.18033
.28000	.51457	.49919	.46890	.46040	.45140	.44152	.43081	.41911	.40637	.39243	.37716	.36036
.42000	.72960	.71351	.67713	.66662	.65528	.64278	.62911	.61409	.59763	.57953	.55960	.53755
.56000	.91644	.90330	.86581	.85451	.84211	.82831	.81308	.79621	.77759	.75699	.73416	.70877
.70000	1.07578	1.06835	1.03376	1.02265	1.01026	.99628	.98067	.96323	.94380	.92213	.89795	.87087
.84000	1.20839	1.20913	1.18043	1.17028	1.15871	1.14544	1.13042	1.11342	1.09428	1.07274	1.04850	1.02114
.98000	1.31888	1.32681	1.30597	1.29732	1.28717	1.27527	1.26155	1.24578	1.22779	1.20731	1.18403	1.15752
1.12000	1.40552	1.42315	1.41128	1.40445	1.39610	1.38600	1.37406	1.36006	1.34384	1.32510	1.30355	1.27874
1.26000	1.47411	1.50037	1.49779	1.49295	1.48560	1.47853	1.46865	1.45675	1.44266	1.42611	1.40679	1.38428
1.40000	1.52673	1.56096	1.56741	1.56456	1.56025	1.55428	1.54657	1.53691	1.52514	1.51100	1.49419	1.47432
1.54000	1.56618	1.60747	1.62227	1.62131	1.61896	1.61504	1.60945	1.60202	1.59259	1.58091	1.56671	1.54961
1.68000	1.59511	1.64240	1.66460	1.66537	1.66481	1.66278	1.65918	1.65386	1.64665	1.63736	1.62570	1.61134
1.82000	1.61583	1.66806	1.69658	1.69886	1.69988	1.69953	1.69772	1.69430	1.68914	1.68204	1.67275	1.66098
1.96000	1.63032	1.68650	1.72025	1.72379	1.72616	1.72726	1.72699	1.72524	1.72187	1.71672	1.70957	1.70014
2.10000	1.64022	1.69945	1.73738	1.74197	1.74545	1.74774	1.74877	1.74842	1.74659	1.74312	1.73781	1.73043
2.24000	1.64683	1.70835	1.74954	1.75495	1.75932	1.76258	1.76466	1.76547	1.76490	1.76282	1.75907	1.75344
2.38000	1.65113	1.71432	1.75797	1.76402	1.76908	1.77310	1.77601	1.77774	1.77819	1.77725	1.77478	1.77058
2.52000	1.65387	1.71824	1.76371	1.77023	1.77582	1.78041	1.78397	1.78641	1.78766	1.78762	1.78616	1.78312
2.66000	1.65556	1.72076	1.76752	1.77439	1.78036	1.78539	1.78943	1.79241	1.79427	1.79492	1.79425	1.79212
2.80000	1.65659	1.72233	1.77000	1.77712	1.78337	1.78872	1.79310	1.79648	1.79879	1.79996	1.79989	1.79846
2.94000	1.65719	1.72329	1.77158	1.77889	1.78532	1.79089	1.79552	1.79919	1.80183	1.80338	1.80375	1.80284
3.08000	1.65754	1.72387	1.77256	1.77998	1.78656	1.79227	1.79709	1.80096	1.80383	1.80565	1.80635	1.80582
3.22000	1.65773	1.72421	1.77317	1.78066	1.78733	1.79315	1.79808	1.80209	1.80512	1.80714	1.80806	1.80780
3.36000	1.65784	1.72440	1.77352	1.78107	1.78779	1.79368	1.79869	1.80279	1.80594	1.80809	1.80916	1.80910
3.50000	1.65789	1.72450	1.77373	1.78131	1.78807	1.79400	1.79907	1.80323	1.80645	1.80868	1.80987	1.80993
3.64000	1.65792	1.72456	1.77385	1.78145	1.78823	1.79419	1.79929	1.80349	1.80676	1.80905	1.81030	1.81046
3.78000	1.65794	1.72459	1.77392	1.78153	1.78833	1.79430	1.79942	1.80364	1.80694	1.80927	1.81057	1.81078
3.92000	1.65795	1.72461	1.77396	1.78157	1.78838	1.79436	1.79949	1.80373	1.80705	1.80940	1.81073	1.81098
4.06000	1.65795	1.72462	1.77398	1.78159	1.78841	1.79440	1.79953	1.80378	1.80711	1.80948	1.81082	1.81110
4.20000	1.65795	1.72462	1.77399	1.78160	1.78842	1.79442	1.79956	1.80381	1.80715	1.80952	1.81088	1.81117
4.34000	1.65795	1.72462	1.77399	1.78161	1.78843	1.79443	1.79957	1.80383	1.80717	1.80954	1.81091	1.81121
4.48000	1.65795	1.72462	1.77399	1.78161	1.78843	1.79443	1.79958	1.80384	1.80718	1.80956	1.81093	1.81123
4.62000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79443	1.79958	1.80384	1.80718	1.80956	1.81094	1.81124
4.76000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79443	1.79958	1.80384	1.80719	1.80957	1.81094	1.81125
4.90000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81094	1.81125
5.04000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81125
5.18000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81125
5.32000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
5.46000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
5.60000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
5.74000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
5.88000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
6.02000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
6.16000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
6.30000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
6.44000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
6.58000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
6.72000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
6.86000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
7.00000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126

```
NCYCLE        7          7          7          8          8          9          10         12

R+ (RAD)=  1.39626    1.41372    1.43117    1.44862    1.46608    1.48353    1.50098    1.51844
   (DEG)= 80.00000   81.00000   82.00000   83.00000   84.00000   85.00000   86.00000   87.00000

 0.00000   0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
  .14000    .16987     .15823     .14516     .13024     .11286     .09190     .06510     .02541
  .28000    .34176     .32100     .29758     .27073     .23926     .20107     .15174     .07719
  .42000    .51304     .48554     .45437     .41846     .37613     .32438     .25681     .15256
  .56000    .68036     .64835     .61186     .56958     .51943     .45766     .37613     .24779
  .70000    .84039     .80582     .76619     .72000     .66485     .59639     .50507     .35858
  .84000    .99012     .95471     .91384     .86590     .80926     .73615     .63897     .48030
  .98000   1.12723    1.09237    1.05186    1.00400     .94604     .87295     .77348     .60838
 1.12000   1.25013    1.21693    1.17803    1.13173    1.07525    1.00344     .90477     .73850
 1.26000   1.35803    1.32728    1.29093    1.24732    1.19370    1.12498    1.02967     .86679
 1.40000   1.45085    1.42306    1.38989    1.34976    1.30000    1.23572    1.14576     .99000
 1.54000   1.52912    1.50454    1.47491    1.43871    1.39345    1.33451    1.25132    1.10556
 1.68000   1.59383    1.57252    1.54653    1.51445    1.47400    1.42090    1.34536    1.21158
 1.82000   1.64631    1.62815    1.60572    1.57773    1.54212    1.49501    1.42751    1.30687
 1.96000   1.68806    1.67283    1.65372    1.62961    1.59865    1.55741    1.49793    1.39066
 2.10000   1.72056    1.70804    1.69194    1.67137    1.64473    1.60900    1.55722    1.46353
 2.24000   1.74564    1.73528    1.72181    1.70439    1.68162    1.65090    1.60624    1.52527
 2.38000   1.76443    1.75597    1.74475    1.73003    1.71063    1.68434    1.64608    1.57683
 2.52000   1.77830    1.77141    1.76204    1.74959    1.73305    1.71058    1.67792    1.61915
 2.66000   1.78836    1.78272    1.77485    1.76426    1.75009    1.73082    1.70293    1.65332
 2.80000   1.79552    1.79085    1.78418    1.77506    1.76281    1.74617    1.72227    1.68046
 2.94000   1.80052    1.79660    1.79085    1.78289    1.77216    1.75762    1.73697    1.70167
 3.08000   1.80395    1.80060    1.79554    1.78847    1.77891    1.76603    1.74797    1.71799
 3.22000   1.80627    1.80332    1.79877    1.79237    1.78370    1.77210    1.75607    1.73036
 3.36000   1.80780    1.80514    1.80097    1.79505    1.78705    1.77641    1.76195    1.73958
 3.50000   1.80880    1.80635    1.80244    1.79687    1.78935    1.77942    1.76614    1.74635
 3.64000   1.80943    1.80712    1.80340    1.79808    1.79091    1.78150    1.76908    1.75125
 3.78000   1.80983    1.80762    1.80402    1.79887    1.79194    1.78290    1.77112    1.75475
 3.92000   1.81008    1.80792    1.80441    1.79938    1.79262    1.78384    1.77251    1.75720
 4.06000   1.81022    1.80811    1.80465    1.79970    1.79306    1.78445    1.77344    1.75890
 4.20000   1.81031    1.80823    1.80480    1.79990    1.79333    1.78485    1.77405    1.76006
 4.34000   1.81036    1.80829    1.80489    1.80002    1.79350    1.78510    1.77445    1.76084
 4.48000   1.81039    1.80833    1.80494    1.80009    1.79361    1.78526    1.77471    1.76136
 4.62000   1.81041    1.80836    1.80498    1.80014    1.79367    1.78536    1.77487    1.76169
 4.76000   1.81042    1.80837    1.80499    1.80016    1.79371    1.78541    1.77497    1.76191
 4.90000   1.81042    1.80837    1.80500    1.80018    1.79373    1.78545    1.77504    1.76205
 5.04000   1.81043    1.80839    1.80501    1.80019    1.79374    1.78547    1.77507    1.76214
 5.18000   1.81043    1.80839    1.80501    1.80019    1.79375    1.78548     .77510    1.76219
 5.32000   1.81043    1.80838    1.80501    1.80019    1.79376    1.78549    1.77511    1.76222
 5.46000   1.81043    1.80838    1.80501    1.80019    1.79376    1.78549    1.77512    1.76224
 5.60000   1.81043    1.80839    1.80501    1.80019    1.79376    1.78550    1.77512    1.76225
 5.74000   1.81043    1.80839    1.80501    1.80020    1.79376    1.78550    1.77512    1.76226
 5.88000   1.81043    1.80839    1.80501    1.80020    1.79376    1.78550    1.77512    1.76226
 6.02000   1.81043    1.80838    1.80501    1.80020    1.79376    1.78550    1.77513    1.76227
 6.16000   1.81043    1.80839    1.80501    1.80020    1.79376    1.78550    1.77513    1.76227
 6.30000   1.81043    1.80838    1.80501    1.80020    1.79376    1.78550    1.77513    1.76227
 6.44000   1.81043    1.80838    1.80501    1.80020    1.79376    1.78550    1.77513    1.76227
 6.58000   1.81043    1.80839    1.80501    1.80020    1.79376    1.78550    1.77513    1.76227
 6.72000   1.81043    1.80838    1.80501    1.80020    1.79376    1.78550    1.77513    1.76227
 6.86000   1.81043    1.80839    1.80501    1.80020    1.79376    1.78550    1.77513    1.76227
 7.00000   1.81043    1.80839    1.80501    1.80020    1.79376    1.78550    1.77513    1.76227

DGAMMA=    1.552735     SMALLM=     .035059
```

POINT VORTEX LOCATIONS

TOP BOUNDARY LAYER | BOTTOM BOUNDARY LAYER

A	K	X(A)	Y(A)	XDOT(A)	YDOT(A)	GAMMA(A)	XB(A)	YB(A)	XDOTB(A)	YDOTE(A)	GMAB(A)
1	39	-1.470798	.387896	-.084505	-.001953	.017547	-1.470798	-.387896	-.084505	.001953	-.017547
2	39	-1.478492	.562271	-.400360	-.037539	.037135	-1.478492	-.562271	-.400360	.037539	-.037135
3	39	-1.374651	.513777	-.274386	-.037525	.061020	-1.374651	-.513777	-.274386	.037525	-.061020
4	39	-1.204024	.592420	-.283578	.054769	.072364	-1.204024	-.592420	-.283578	-.054769	-.072364
5	39	-1.250174	.423280	-.092518	.084677	.084417	-1.250174	-.423280	-.092518	-.084677	-.084417
6	39	-1.049625	.658744	-.317442	.186665	.080703	-1.049625	-.658744	-.317442	-.186665	-.080703
7	39	-.897825	.456571	.231030	.491555	.095220	-.897825	-.456571	.231030	-.491555	-.095220
8	39	-1.173107	.302748	.129747	.263879	.106676	-1.173107	-.302748	.129747	-.263879	-.106676
9	39	-1.011672	.495365	-.028465	.268545	.102046	-1.011672	-.495365	-.028465	-.268545	-.102046
10	39	-.915831	.436608	.237372	.536720	.115301	-.915831	-.436608	.237372	-.536720	-.115301
11	39	-1.007367	.195213	.115211	.471943	.124202	-1.007367	-.195213	.115211	-.471943	-.124202
12	39	-1.673170	.311740	-.007193	-.031816	.131124	-1.673170	-.311740	-.007193	.031816	-.131124
13	39	-1.094772	.049451	.405832	.028246	.126061	-1.094772	-.049451	.405832	-.028246	-.126061
14	39	-1.499694	.161685	.392592	.037173	.134868	-1.499694	-.161685	.392592	-.037173	-.134868
15	39	-1.319320	.198246	.317274	.082329	.129898	-1.319320	-.198246	.317274	-.082329	-.129898
16	39	-1.607210	.030250	1.050349	-.005364	.127849	-1.607210	-.030250	1.050349	.005364	-.127849
17	39	-2.076149	.255191	-.185664	-.474283	.135813	-2.076149	-.255191	-.135664	.474283	-.135813
18	39	-1.710810	.148758	.450861	-.114611	.131037	-1.710810	-.148758	.450361	.114611	-.131037
19	39	-1.983040	.186633	.276527	-.098828	.129110	-1.983040	-.186633	.276527	.098328	-.129110
20	39	-2.204532	.125398	.014800	-.275954	.127702	-2.204532	-.125398	.014800	.275954	-.127702
21	39	-1.624535	.627607	-.481717	-.082781	.134314	-1.624535	-.627607	-.481717	.082781	-.134314
22	39	-1.795804	.462980	-.345550	-.150479	.129791	-1.795804	-.462980	-.345550	.150479	-.129791
23	39	-1.940624	.367292	-.170558	-.338341	.128055	-1.940624	-.367292	-.170558	.338341	-.128055
24	39	-1.961811	.481517	-.641739	-.599320	.126807	-1.961811	-.481517	-.641739	.599320	-.126807
25	39	-1.864390	.601270	-.847009	-.472699	.125738	-1.864390	-.601270	-.847009	.472699	-.125738
26	39	-1.666043	.723831	-.902404	-.217858	.124786	-1.666043	-.723831	-.902404	.217858	-.124786
27	39	-1.728547	.794835	-1.210554	-.499095	.123862	-1.728547	-.794835	-1.210554	.499095	-.123862
28	39	-1.294580	.767692	-.569317	-.130324	.124797	-1.294580	-.767692	-.569317	.130324	.124797
29	39	-1.398426	.819677	-.948762	-.372662	.123000	-1.398426	-.819677	-.948762	.372662	-.123000
30	39	-1.221233	.865369	-.945012	-.188080	.121990	-1.221233	-.865369	-.945012	.188080	-.121990
31	39	-1.127894	.909917	-1.034864	-.050010	.121092	-1.127894	-.909917	-1.034864	.050010	-.121092
32	39	-.970191	.899408	-.728856	-.174231	.122041	-.970191	-.899408	-.728856	.174231	-.122041
33	39	-.880088	.931829	-.855026	-.078367	.120286	-.880088	-.931829	-.855026	.078367	-.120286
34	39	-.769794	.949858	-.836941	-.104686	.119314	-.769794	-.949858	-.836941	.104686	-.119314
35	39	-.654144	.970624	-.894004	-.135379	.118474	-.654144	-.970624	-.894004	.135379	-.118474
36	39	-.523922	.997195	-1.025793	-.180170	.117752	-.523922	-.997195	-1.025793	.180170	-.117752
37	39	-.353272	1.030128	-1.257200	-.240873	.118833	-.353272	-1.030128	-1.257200	.240873	-.118633
38	39	-.159547	1.048199	-1.529307	-.134673	.117236	-.159547	-1.048199	-1.529307	.134673	-.117236
39	39	.054171	1.033641	-1.704354	.115828	.116460	.054171	-1.033641	-1.704354	-.115828	-.116460

TOP REAR SHEAR LAYER | BOTTOM REAR SHEAR LAYER

A	K	XRT(A)	YRT(A)	XROOT(A)	YROOT(A)	GMRT(A)	XRB(A)	YRB(A)	XROOTB(A)	YROOTB(A)	GMAB(A)
1	39	-1.391255	.428310	-.060685	-.034127	-.001701	-1.391255	-.428310	-.060685	.034127	.001701
2	39	-1.366517	.602144	-.457523	-.084011	-.009489	-1.366517	-.602144	-.457523	.084011	.009489
3	39	-1.445651	.649619	-.541731	-.076439	-.020866	-1.445651	-.649619	-.541731	.076439	.020866
4	39	-.960804	.712634	-.271053	.221530	-.012567	-.960804	-.712634	-.271053	-.221530	.012567
5	39	-.850305	.590171	.030460	.210316	-.064597	-.850305	-.590171	.030460	-.210316	.064597

```
                                   PRESSURE DISTRIBUTION
              T= 5.0477       ZHAT=  .8770      AK= 1.0000      AKDOT= 0.0000       THETAS=  87.00

   DEG          PHIVT         PHIPT          2(PHIT)        PSTKSQD        CPK           PDRAG          UTAN
    0.00        .007121      0.000000        .014242        -.000000       .014242       .014242       -.000000
   10.00        .007059      0.000000        .014119        -.115164      -.101046      -.099511        .339359
   20.00        .006851      0.000000        .013701        -.445701      -.432000      -.405947        .667608
   30.00        .006413      0.000000        .012826        -.948413      -.935586      -.810241        .973865
   40.00        .005566      0.000000        .011131       -1.556624     -1.545493     -1.183916       1.247647
   50.00        .003913      0.000000        .007826       -2.187262     -2.179436     -1.400915       1.478939
   60.00        .000524      0.000000        .001048       -2.748803     -2.747755     -1.373878       1.657952
   65.00       -.002488      0.000000       -.004975       -2.974329     -2.979304     -1.259108       1.724624
   70.00       -.007181      0.000000       -.014366       -3.147067     -3.161433     -1.081274       1.773997
   75.00       -.014934      0.000000       -.029868       -3.253856     -3.283724      -.849890       1.803845
   80.00       -.028889      0.000000       -.057779       -3.277652     -3.335431      -.579192       1.810429
   85.00       -.057824      0.000000       -.115648       -3.188001     -3.303649      -.287932       1.785498
   90.00       -.135008      0.000000       -.270016       -2.898292     -3.168308      -.000000       1.702437
   95.00       -.457438      0.000000       -.914875       -2.016878     -2.931754       .255519       1.420168
  100.00       -.954251      0.000000      -1.908499       -1.043036     -2.951535       .512529       1.021291
  105.00       -.595690      0.000000      -1.191379       -1.270289     -2.461668       .637127       1.127071
  110.00       -.780841      0.000000      -1.561682        -.704720     -2.266402       .775155        .839476
  115.00       -.673573      0.000000      -1.347147        -.574824     -1.921971       .812260        .758172
  120.00       -.730796      0.000000      -1.461592        -.302666     -1.764258       .882129        .550151
  125.00       -.745331      0.000000      -1.490663        -.164166     -1.654829       .949171        .405174
  130.00       -.758687      0.000000      -1.517375        -.077875     -1.595249      1.025406        .279060
  135.00       -.766323      0.000000      -1.532646        -.027253     -1.559899      1.103015        .165085
  140.00       -.775455      0.000000      -1.550910        -.005173     -1.556083      1.192029        .071922
POTENTIAL VORTEX APPROX INVALID   ALPHA=  5      CORE RADIUS=      .035297
  145.00       -.715488      0.000000      -1.430977        -.037699     -1.468676      1.203069       -.194163
  150.00       -.543223      0.000000      -1.086447        -.234987     -1.321433      1.144395       -.484754
POTENTIAL VORTEX APPROX INVALID   ALPHA=  7      CORE RADIUS=      .034997
POTENTIAL VORTEX APPROX INVALID   ALPHA= 10      CORE RADIUS=      .016923
  155.00       -.752162      0.000000      -1.504324        -.359233     -1.863557      1.688956       -.599360
  160.00       -.709889      0.000000      -1.419778        -.424160     -1.843937      1.732734       -.651276
  165.00       -.686461      0.000000      -1.372923        -.515227     -1.888150      1.823813       -.717793
POTENTIAL VORTEX APPROX INVALID   ALPHA= 11      CORE RADIUS=      .031208
  170.00       -.756222      0.000000      -1.512445        -.233699     -1.746144      1.719616       -.483425
  175.00       -.652105      0.000000      -1.304209        -.253229     -1.557438      1.551512       -.503219
  180.00       -.551320      0.000000      -1.102640        -.000000     -1.102640      1.102640       -.000000


K= 39
CDPK=      .503871
CDSK=      .012166
 CGK=      .516037
 CDN=      .034568
```

```
MAX VELOCITY= 1.811 AT THETA= 79.0 DEGREES
MAX BACKFLOW VEL= -.718 AT THETA=165.0 DEGREES
MIN TANGENTIAL VELOCITY BETWEEN MAX VELOCITY AND MAX BACKFLOW VELOCITY=  .072 AT THETA=140.0 DEGREES
REAR SEPARATION ANGLE=161.7 DEGREES

RVTX SUBROUTINE N=  5  THETAS(N)=  161.721311  U00(N)=    -.635553
GAMA(N)=    -.02524

GAMA CHECK SUM=   -.058117    N=  5      IFLAG(N-1)=  1

BEFORE VMFIX(X(ALPHA),Y(ALPHA))=( -.85706,  .50642)

VMFIX  ALPHA= 10  ANGLE=  -75.488734
X=    -.903573  Y=      .430764  XDOT=      .304025  YDOT=      .637725

AFTER  VMFIX(X(ALPHA),Y(ALPHA))=( -.86557,  .51048)
ELAPSED TIME=   44.988000
```

```
      ADYNF      SAMPLE      INPUT
      CASE ... OGIVE CYLINDER ... ANGLE OF ATTACK (5 DEG)


    *****     ADYNF   SAMPLE   CASE       *****                                      CARD 1
5.0         0.0          50.478       10.74        2.35         2.35                 CARD 2
  4  16                                                                              CARD 3
0.0         0.0                                                                      CARD 4
    .096516     9.036431     .181378  1                                              CARD 5
    .163031     8.165342     .306378  2                                              CARD 6
    .229547     4.086353     .431378  3                                              CARD 7
    .296063      .047975     .556378  4                                              CARD 8
    .362578      .041644     .681378  5                                              CARD 9
    .429094      .043186     .806378  6                                              CARD 10
    .495610      .055931     .931378  7                                              CARD 11
    .562125      .069124    1.056378  8                                              CARD 12
    .628641      .087352    1.181378  9                                              CARD 13
    .695157      .105378    1.306378 10                                              CARD 14
    .761672      .132163    1.431378 11                                              CARD 15
    .828188      .154608    1.556378 12                                              CARD 16
    .894704      .187945    1.681378 13                                              CARD 17
    .961219      .222631    1.806378 14                                              CARD 18
   1.0           .249663    1.931378 15                                              CARD 19
0.0          .125         .3          1.0                                            CARD 20
```

```
                    ***** ADYNF  SAMPLE  OUTPUT *****

GIVEN DATA

   1    0.00000000     0.00000000
   2     .09651600      .06864185
   3     .16303100      .06202495
   4     .22954700      .03104044
   5     .29606300      .00036442
   6     .36257800      .00031633
   7     .42909400      .00032805
   8     .49561000      .00042486
   9     .56212500      .00052507
  10     .62864100      .00066354
  11     .69515700      .00080046
  12     .76167200      .00100393
  13     .82818800      .00117442
  14     .89470400      .00142765
  15     .96121900      .00169113
  16    1.00000000      .00189647

NO. OF INITIAL          KNOTS =  4
ITER =  8
KNOTS PRIOR TO OPTIMIZATION
   0.000000     .125000     .300000    1.000000
                                               *** FINAL OUTPUT ***


                                          KNOTS                    CUBIC COEFFICIENTS


                                     XI( 1) =     0.000000
                                                                  C(1) =    -2.892801E-14
                                                                  C(2) =    -4.355484E+00
                                                                  C(3) =     1.185283E+02
                                                                  C(4) =     6.991980E+02
                                     XI( 2) =      .071209
                                                                  C(1) =     3.840772E-02
                                                                  C(2) =     1.888742E+00
                                                                  C(3) =    -3.083995E+01
                                                                  C(4) =     1.349466E+02
                                     XI( 3) =      .149811
                                                                  C(1) =     6.186244E-02
                                                                  C(2) =    -4.582100E-01
                                                                  C(3) =     9.812087E-01
                                                                  C(4) =    -6.308512E-01
                                     XI( 4) =     1.000000


                                         LEAST SQUARE ERROR =         4.933780E-03
                                         AVERAGE ERROR      =         3.812333E-03
                                         MAXIMUM ERROR      =         1.349815E-02 AT    .296063
```

```
                         APPROXIMATION AND SCALED ERROR CURVE
                      DATA POINT         APPROXIMATION   DEVIATION X 10E+3
                 1    0.00000000         -.00000000           .000000
                 2     .09551600          .06864185           .000000
                 3     .16303100          .05597500          6.049947
                 4     .22954700          .03124520          -.204763
                 5     .29606300          .01386257        -13.496148
                 6     .36257800          .00271330         -2.396967
                 7     .42909400         -.00331686          3.644905
                 8     .49561000         -.00534162          5.766478
                 9     .56212500         -.00447494          5.000012
                10     .62864100         -.00183070          2.494236
                11     .69515700          .00147717          -.676710
                12     .76167200          .00433472          3.330796
                13     .82818800          .00562812         -4.453697
                14     .89470400          .00424339         -2.815733
                15     .96121900         -.00093329          2.624420
                16    1.00000000         -.00614404          8.040511

      Z               CON(Z)
   0.000000         -.000000
    .050000         -.008853
    .100000          .070443
    .150000          .061776
    .200000          .041257
    .250000          .025170
    .300000          .013040
    .350000          .004395
    .400000         -.001238
    .450000         -.004332
    .500000         -.005361
    .550000         -.004798
    .600000         -.003116
    .650000         -.000787
    .700000          .001714
    .750000          .003915
    .800000          .005342
    .850000          .005523
    .900000          .003985
    .950000          .000253
   1.000000         -.006144
 BEG,STEP   0.                   1.00000000E+00     1
 BEG,STEP   5.00000000E-01  5.00000000E-01     1
 H2 CONVERGENCE AT ROW  3
 INTEGRAL IS  3.46173213E-04, ERROR 2.01407345E-04  FROM T(3,2)
 BEG,STEP   0.                   5.00000000E-01     1
 BEG,STEP   2.50000000E-01  2.50000000E-01     1
 H2 CONVERGENCE AT ROW  3
 INTEGRAL IS  1.03065782E-03, ERROR 9.03363896E-05  FROM T(3,2)
 BEG,STEP   0.                   2.50000000E-01     1
 H2 CONVERGENCE AT ROW  5
 INTEGRAL IS  8.09074826E-03, ERROR 8.07258925E-05  FROM T(5,2)


 ..... CN=      .129465
 ..... CNI=      .009468      ERROR=   .00037247     IFLAG= 1
```

```
GIVEN DATA

  1     0.00000000    0.00000000
  2      .09651600     .00702654
  3      .16303100     .01043383
  4      .22954700     .00720532
  5      .29606300     .00010789
  6      .36257800     .00011470
  7      .42909400     .00014076
  8      .49561000     .00021056
  9      .56212500     .00029516
 10      .62864100     .00041713
 11      .69515700     .00055645
 12      .76167200     .00076466
 13      .82818800     .00097264
 14      .89470400     .00127733
 15      .96121900     .00162555
 16     1.00000000     .00189647

NO. OF INITIAL          KNOTS =  4
ITER =  8



KNOTS PRIOR TO OPTIMIZATION
   0.000000     .071209     .149811    1.000000
                                        ***  FINAL OUTPUT  ***


                              KNOTS                      CUBIC COEFFICIENTS


                         XI( 1) =     0.000000
                                                     C(1) =   -1.284775E-14
                                                     C(2) =   -9.314970E+00
                                                     C(3) =    2.089747E+02
                                                     C(4) =   -1.183289E+03
                         XI( 2) =      .071209
                                                     C(1) =   -3.091969E-02
                                                     C(2) =    2.446370E+00
                                                     C(3) =   -4.380895E+01
                                                     C(4) =    2.525566E+02
                         XI( 3) =      .129259
                                                     C(1) =    1.286898E-02
                                                     C(2) =   -8.665235E-02
                                                     C(3) =    1.733354E-01
                                                     C(4) =   -1.033547E-0[illegible]
                         XI( 4) =     1.000000


                            LEAST SQUARE ERROR =        9.302876E-04
                            AVERAGE ERROR      =        6.924194E-04
                            MAXIMUM ERROR      =        2.650252E-03 AT     .296063
```

```
                                          APPROXIMATION AND SCALED ERROR CURVE
                                    DATA POINT          APPROXIMATION    DEVIATION X 10E+4
                                1    0.00000000          -.00000000          -.000000
                                2     .09651600           .00702654          -.000000
                                3     .16303100           .01013625          3.025774
                                4     .22954700           .00581768         13.884436
                                5     .29606300           .00275814        -26.502519
                                6     .36257800           .00077458         -6.598836
                                7     .42909400          -.00031538          4.561377
                                8     .49561000          -.00069418          9.047434
                                9     .56212500          -.00054434          8.394947
                               10     .62864100          -.00004834          4.654675
                               11     .69515700           .00061131          -.548631
                               12     .76167200           .00125211         -4.874507
                               13     .82818800           .00169159         -7.189456
                               14     .89470400           .00174722         -4.698974
                               15     .96121900           .00123654          3.890096
                               16    1.00000000           .00060493         12.915449
      Z             (.5-Z)CON
   0.000000          -.000000
    .050000          -.091223
    .100000           .009227
    .150000           .011145
    .200000           .007570
    .250000           .004752
    .300000           .002613
    .350000           .001076
    .400000           .000063
    .450000          -.000502
    .500000          -.000699
    .550000          -.000603
    .600000          -.000293
    .650000           .000154
    .700000           .000661
    .750000           .001149
    .800000           .001542
    .850000           .001761
    .900000           .001730
    .950000           .001370
   1.000000           .000605
 BEG,STEP   0.                1.00000000E+00     1
 BEG,STEP   5.00000000E-01   5.00000000E-01     1
 H2 CONVERGENCE AT ROW   3
 INTEGRAL IS  3.75214479E-04, ERROR 2.49143177E-05   FROM T(3,2)
 BEG,STEP   0.                5.00000000E-01     1
 BEG,STEP   2.50000000E-01   2.50000000E-01     1
 H2 CONVERGENCE AT ROW   3
 INTEGRAL IS  2.53645229E-04, ERROR 1.58105198E-05   FROM T(3,2)
 BEG,STEP   0.                2.50000000E-01     1
 H2 CONVERGENCE AT ROW   5
 INTEGRAL IS -4.86222290E-03, ERROR 1.84561813E-04   FROM T(5,2)
```

```
 ..... CM=       .057890
 ..... CMI=     -.004233      ERROR=    .00022529      IFLAG= 1
```

LIST OF REFERENCES

1. Hall, M. G., "A Numerical Method for Calculating the Unsteady Two Dimensional Laminar Boundary Layer," Ing-Arch 38 97 (1969).

2. Wundt, H., "Wach stum der laminanen Grenzschicht an schräg ange strömten Zylinder bei Anfahrt ans der Ruhe," Ingen-Arch 23 212 (1955).

3. Carl de Boor and John R. Rice, "Least Squares Cubic Spline Approximation I - Fixed Knots," Purdue University Computer Sciences Department Report CSD TR20, (1968).

4. Carl de Boor and John R. Rice, "Least Squares Cubic Spline Approximation II - Variable Knots," Purdue University Computer Sciences Department Report CSD TR21, (1968).

☆ U.S. GOVERNMENT PRINTING OFFICE: 1974—739-161/138